Trabajo Fin de Máster
Máster en Diseño Avanzado en Ingeniería Mecánica

# Observación de Estados de un Vehículo a Control Remoto Mediante Modelo de Sistema Multicuerpo, Medidas Inerciales y Visión Artificial

Autor: Pedro Urda Gómez

Tutor: Sergio Muñoz Moreno

**Dpto. Ingeniería Mecánica y Fabricación**
**Escuela Técnica Superior de Ingeniería**
**Universidad de Sevilla**

Sevilla, 2018

Trabajo Fin de Máster
Máster en Diseño Avanzado en Ingeniería Mecánica

# Observación de Estados de un Vehículo a Control Remoto Mediante Modelo de Sistema Multicuerpo, Medidas Inerciales y Visión Artificial


Autor:

Pedro Urda Gómez

Tutor:

Sergio Muñoz Moreno

Profesor titular

Dpto. de Teoría Ingeniería Mecánica y Fabricación

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla


Sevilla, 2018

Trabajo Fin de Máster: **Observación de Estados de un Vehículo a Control Remoto Mediante Modelo de Sistema Multicuerpo, Medidas Inerciales y Visión Artificial**

Autor: Pedro Urda Gómez
Tutor: Sergio Muñoz Moreno

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

# Índice

# Índice de Figuras

# Índice de Tablas

# Capítulo 1

# Introducción

En este Trabajo Fin de Máster se ha llevado a cabo la construcción de un modelo de sistema multicuerpo de un vehículo eléctrico a control remoto que de aquí en adelante será denominado *Róver*. Dicho vehículo ha sido equipado con una serie de sensores inerciales que permitirán recoger medidas experimentales para validar el modelo de sistema multicuerpo construido. También se ha programado un algoritmo de visión artificial que permite validar la trayectoria estimada por el modelo.

## 1.1. Antecedentes

El vehículo a control remoto objeto fundamental de este TFM denominado como *Róver*, fue resultado de un proyecto previo del Departamento de Ingeniería Mecánica de la Universidad de Sevilla.

La idea original era la de construir un vehículo a control remoto, sensorizarlo y emplearlo para la auscultación experimental de superficies. Se pretendía generar algoritmos que permitiesen determinar la geometría vertical de la superficie por la que el vehículo se iba desplazando mediante la utilización de filtro de Kalman y algoritmos de fusión de sensores.

En la figura 1 se muestra una instantánea del diseño original del Róver. Como puede observarse la electrónica del sistema era sumamente rudimentaria. La placa base estaba construida a base de cables soldados sobre una baquelita

perforada. El resultado era un sistema poco fiable debido fundamentalmente a fallos en las conexiones eléctricas.

Tampoco era adecuada la colocación de la IMU, sensor principal del vehículo que más adelante describiremos, que se hallaba desplazada del centro de gravedad del vehículo. Algo subsanable matemáticamente pero absurdo teniendo en cuenta que resulta posible colocarla físicamente mucho más cerca del centro de gravedad (c.d.g) del vehículo.

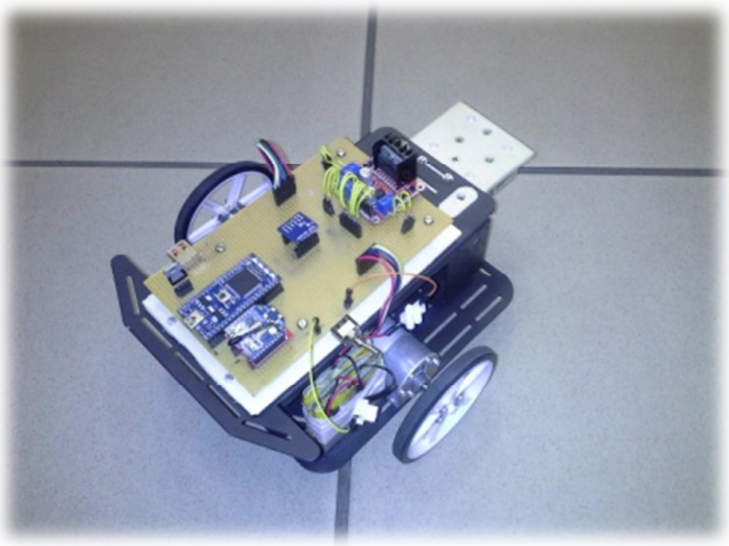


**Figura 1. Construcción original del Róver**

Otro de los problemas que presentaba el vehículo era que la alimentación del sistema se realizaba directamente desde tres baterías *Lipo* conectadas en serie que suministraban una tensión no del todo estable de aproximadamente 13 voltios. Circunstancia igualmente poco deseable para un sistema como tal.

El mando a distancia original presentaba problemas similares a los anteriormente descritos que impedían el correcto funcionamiento del control a distancia.

Vistos estos problemas y la falta de aptitud del sistema para cumplir con su cometido original, se decidió realizar un rediseño y renovación del vehículo que convirtiesen en un auténtico vehículo experimental para la auscultación de superficies.

## 1.2. Descripción general de las modificaciones realizadas en el Róver

Antes de poder realizar cualquier ensayo de auscultación de superficies, era necesario conseguir un vehículo totalmente fiable. Lo primero que se ha hecho ha sido rediseñar desde cero la electrónica del vehículo, introduciéndole notables modificaciones respecto al diseño original.

En el antiguo *Róver* tanto la adquisición de datos, comunicación inalámbrica y control del vehículo se realizaba a través de un único microcontrolador. Concretamente se trataba de una Mbed LPC1768, un pequeño pero potente controlador con procesador Cortex-M3 de 96MHz.

En la nueva versión del *Róver* se ha optado por independizar el control remoto del vehículo de la adquisición.

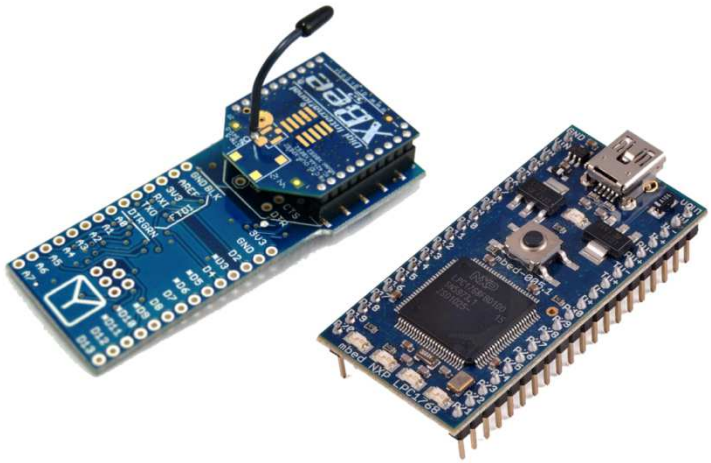


**Figura 2. Controladores Arduino Fio (Izquierda) y Mbed (Derecha)**

Para llevar a cabo el control de los motores y la comunicación inalámbrica se ha instalado en el *Róver* un microcontrolador Arduino Fio. Se trata de un sencillo controlador especialmente diseñado para la comunicación inalámbrica. De esta manera las funciones del Arduino Fio son establecer la comunicación con el mando a distancia, enviar las señales de control apropiadas a los motores y mandar a la Mbed el trigger (señal de disparo) de inicio de la adquisición de datos.

Siguiendo con la descripción de modificaciones introducidas en el vehículo, la IMU de la que más adelante hablaremos, ha sido retirada de su posición original sobre el chasis para ser reubicada en una posición muy próxima al centro de masas del vehículo.

Finalmente, el mando también ha sido rediseñado. Fundamentalmente se ha intentado simplificar al máximo su funcionamiento, suprimiendo la pantalla LCD que montaba originalmente y que lo único que hacía era complicar el sistema. El mando sí que tenía de origen un Arduino Fio con módulo inalámbrico Xbee, el cual obviamente se ha mantenido. Como innovaciones en el diseño del mando se han introducido cuatro botones para el control del vehículo y un Joystick de mayor funcionalidad que el original. Con respecto a la alimentación se ha pasado de las dos baterías Lipo a una pila recargable de 9V.

Una vez conseguido un sistema lo suficientemente fiable se ha construido un modelo del vehículo en MATLAB, con el que se puede simular el comportamiento dinámico del vehículo. Dicho trabajo aparte de ser utilizado en las prácticas de la asignatura Cinemática y Dinámica de Máquinas, pretende ser también útil con fines investigadores.

En las prácticas de Cinemática y Dinámica de Máquinas, los alumnos realizan un experimento de campo con el vehículo en el que recogen datos que luego utilizarán para la validación de un modelo matemático del *Róver* que ellos mismos deben construir durante las prácticas.

Es también de gran interés para el Grupo de Ingeniería Mecánica de la Universidad de Sevilla la auscultación de superficies a partir de medidas de sensores inerciales. Gracias al trabajo desarrollado en este proyecto se dispone ahora de un vehículo equipado con los sensores necesarios para llevar a cabo dicha tarea. La idea es poder utilizar el modelo desarrollado en este TFM en combinación con algoritmos de fusión se sensores y filtrado Kalman para conseguir una lectura precisa de la geometría del terreno por el cual es vehículo se va desplazando.

# Capítulo 2

# Modelo multicuerpo del vehículo

En este segundo capítulo desarrollaremos el modelo multicuerpo del *Róver* que nos permitirá simular su comportamiento cinemático y dinámico a partir de unos inputs sintetizados u obtenidos directamente de los sensores. Para ello se debe primero caracterizar el vehículo identificando todos sus parámetros geométricos másicos e inerciales. Una vez hecho eso se puede proceder a generar el modelo simbólico del vehículo y a partir de él simular su cinemática y su dinámica.

## 2.1 Identificación paramétrica

### 2.1.1 Modelo CAD del vehículo.

Para poder identificar los parámetros másicos e inerciales, en un primer paso se ha procedido a generar un modelo 3D del vehículo en Solidworks a partir del cual obtendremos la posición del centro de gravedad y valores del tensor de inercia del vehículo. En la figura 3 vemos dicho modelo .

**Tabla 1. Masas de los elementos del vehículo**

| Elemento | Masa |
|---|---|
| Motor | 0.178 kg |
| Rueda | 0.022 kg |
| Chasis | 0.328 kg |
| Placa base | 0.170 kg |
| Eje jockey | 0.323 kg |
| Rueda jockey | 0.090 kg |

Con respecto a la posición del centro de gravedad del vehículo, suponiendo que tomamos como referencia el centro de la placa inferior del chasis (color morado) con el eje X alineado en la dirección de avance del vehículo, el eje Z perpendicular al chasis y hacia arriba y el eje Y formando un sistema a derechas con los anteriores, según Solidworks el centro de gravedad del vehículo está ubicado a -32.32 mm en X y a 39.84 mm en Z. Está posición parece a priori totalmente razonable al estar la masa del vehículo mayormente concentrada en la parte trasera del mismo.

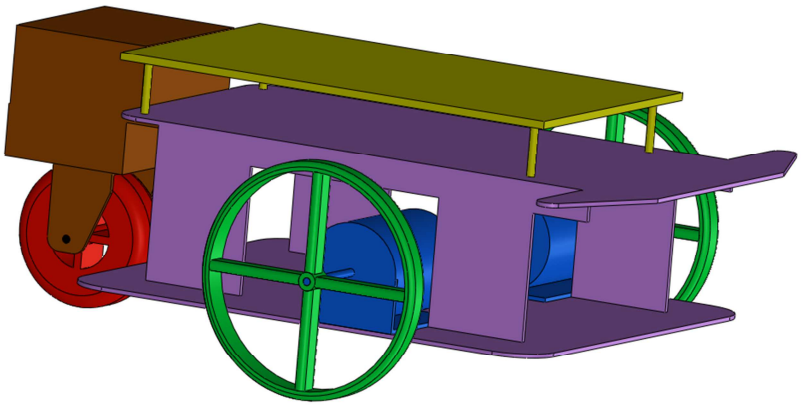


**Figura 3. Modelo CAD del Róver**

El tensor de inercias calculado por Solidworks resulta ser:

$$\bar{\bar{I}} = \begin{bmatrix} 0.0015 & 0 & 0 \\ 0 & 0.001 & 0.0001 \\ 0 & 0.0001 & 0.002 \end{bmatrix} kg \bullet m^2$$

### 2.1.2 Determinación experimental del centro de masas del vehículo

Para cerciorarnos de que los resultados obtenidos son correctos, alternativamente a los cálculos numéricos se han realizado una serie de experimentos para determinar de forma empírica la posición del centro de gravedad del *Róver* y sus momentos principales de inercia.

Supongamos que tenemos un objeto plano con geometría irregular. Si colgado dicho objeto de un punto cualquiera de su superficie este tenderá a orientarse de forma que su centro de gravedad quede alineado con el punto desde el que se ha colgado. Si repetimos dicho experimento colgando el sólido desde un punto distinto volverá a producirse el mismo fenómeno. La intersección de ambas líneas marcará la posición del centro de gravedad. En las figuras siguientes se resume el proceso seguido.

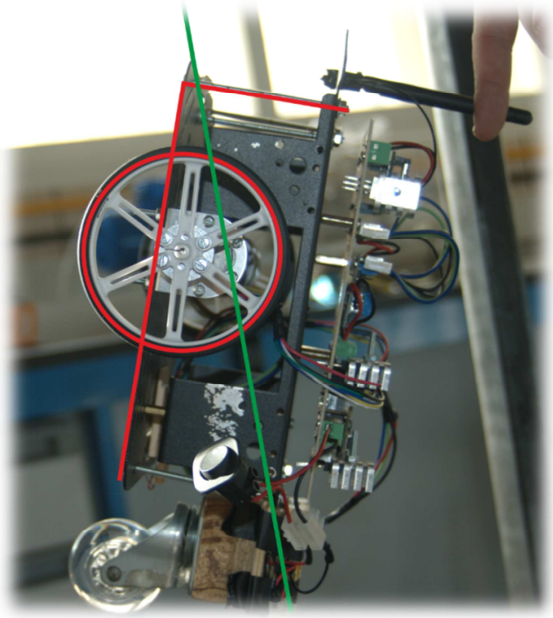


**Figura 4. Determinación del cdg sobre el primer eje**

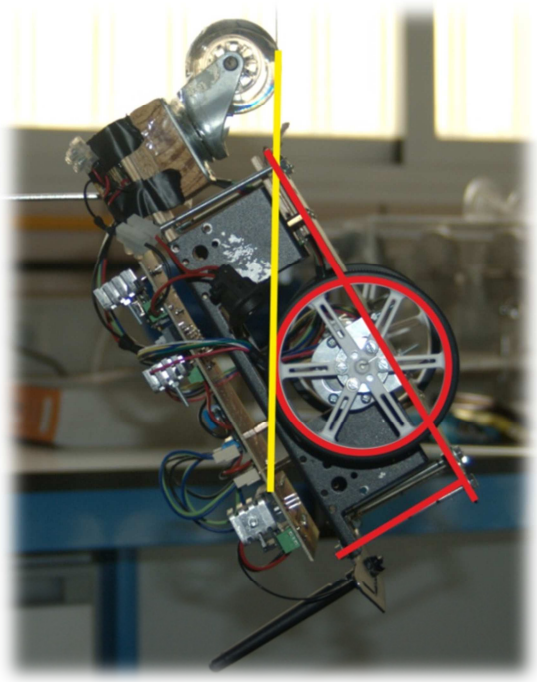


**Figura 5. Determinación del cdg sobre el segundo eje**

Superponiendo ambas imágenes con la escala adecuada llegamos a la construcción geométrica siguiente, donde la intersección de las líneas verde y amarilla determinan la posición del cdg. Tomando como referencia el centro de la rueda y midiendo paralela y perpendicularmente al chasis se obtienen las cotas dx y dz. Aplicando el factor de escala del dibujo se llega a que los valores obtenidos son:

$$dx = -37.67$$
$$dz = 37.8$$

Como puede observarse los valores obtenidos son aproximadamente iguales a los calculador por Solidworks.

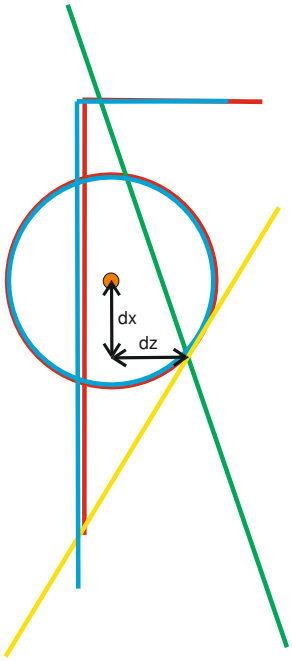



**Figura 6. Determinación experimental del cdg**

### 2.1.3 Determinación experimental del tensor de inercias del vehículo

La determinación experimental del tensor de inercias del vehículo requiere de un proceso algo más sofisticado que el simple cálculo de la posición del centro de masas de un sólido.

Para obtenerlo se debe colgar el vehículo de distintos puntos, hacerlo oscilar en torno a cada uno de los ejes donde se quiere calcular el momento y registrar con el giróscopo la velocidad angular medida en torno a dicho eje. Teniendo en cuenta que durante el experimento la única fuerza que actúa sobre el sistema y que lo hacen pendulear es la gravedad se puede llegar a determinar la inercia polar en torno a ese eje. En la figura 7 vemos como se debe colocar el vehículo para obtener los momentos de inercia polar respecto a cada uno de los ejes. Es fundamental que cuando el vehículo está colgado quede perfectamente alineado con los puntos de colgado, ya que de lo contrario al poner en movimiento el sistema esté se comportaría como un péndulo doble. Recordar que el fenómeno que estamos intentando provocar es un movimiento de péndulo simple en torno a cada uno de los ejes.

Para las dos primeras posiciones representadas en la figura 7, dadas las simetrías del vehículo, se puede conseguir el alineamiento antes comentado, pero sin embargo, para el tercer caso no es posible hacer oscilar el vehículo directamente en torno al eje X del sistema. En este caso el vehículo una vez

colgado queda formando un cierto ángulo respecto al eje X sobre el que nos interesa conocer la medida. Para corregir esto deberemos aplicar la transformación geométrica conveniente desde el eje de rotación hasta el eje X deseado.

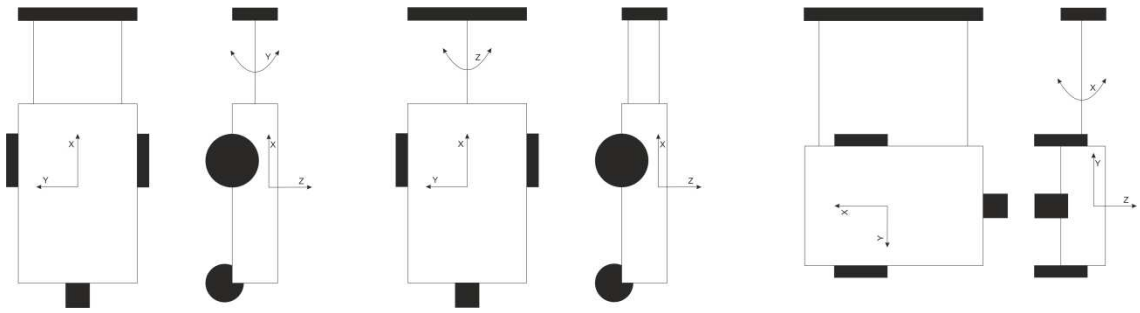


**Figura 7. Determinación experimental de los momentos principales de inercia**

En la figura 8 se muestra el resultado de la adquisición realizada con el giróscopo mientras se ha hecho oscilar el vehículo en torno al eje Z. Como puede observarse la oscilación se va amortiguando con el tiempo y no se ve contaminada por ninguna frecuencia rara. Ello quiere decir que efectivamente el sólido ha oscilado como péndulo simple. La figura 9 se corresponde con respuesta en frecuencia calcula a la señal del giróscopo. A partir de las frecuencias naturales de oscilación obtenidas en cada eje se pueden calcular los momentos principales de inercia con el procedimiento que se describe a continuación.

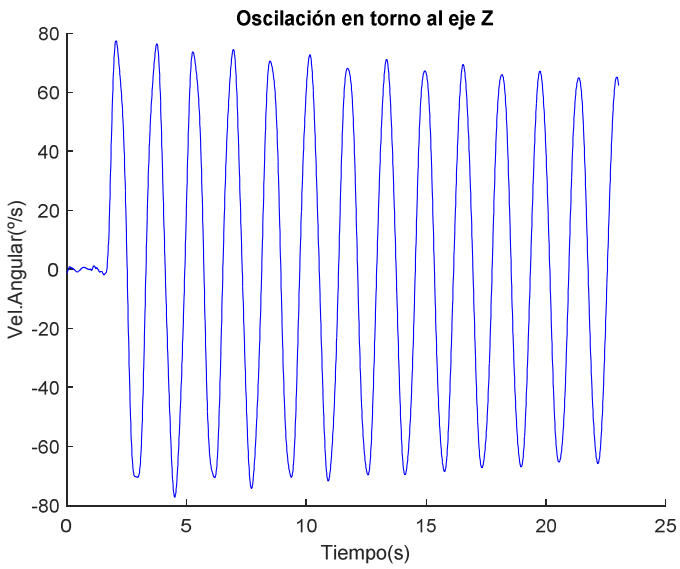



**Figura 8. Oscilación en torno al eje Z**

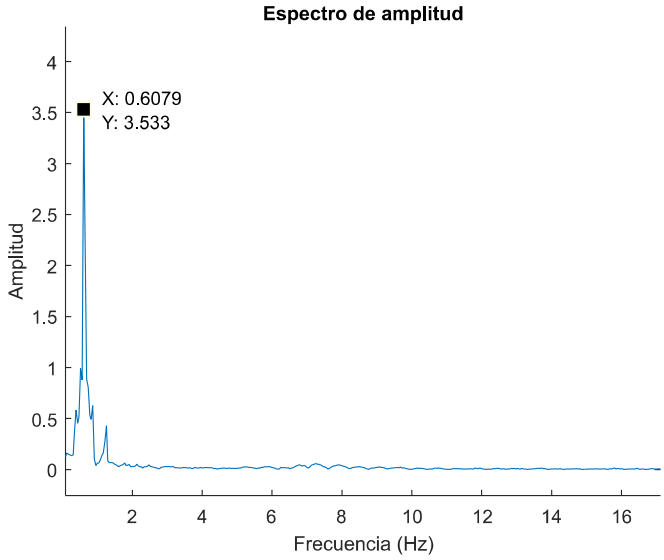



**Figura 9. Respuesta en frecuencia de la señal oscilatoria**

Las ecuaciones siguientes describen en detalle el procedimiento seguido para obtener los momentos polares de inercia en cada eje y a partir de ellos calcular los momentos principales de inercia del vehículo. Nótese que por simplicidad se está asumiendo que todos los productos cruzados de inercia son nulos. Con este desarrollo se pretende obtener una estimación de la magnitud del tensor de inercias del vehículo para poder así comparar con el resultado generado por el software de CAD donde se ha realizado el modelo.

$$\omega_n = \sqrt{\frac{\left(I_G + m \cdot l^2\right)}{m \cdot l \cdot g}} \tag{1}$$

$$I_G = I_z^P = m \cdot l \cdot \left(\frac{g}{\omega_n^{\,2}} - l\right) \tag{2}$$

$$I_z^P = I_{xx} + I_{yy} \tag{3}$$

Procediendo de forma análoga,

$$I_y^P = I_{xx} + I_{zz} \tag{4}$$

Para el caso del eje X, al no producirse la oscilación exactamente en torno a él, es preciso realizar una transformación geométrica desde el eje $X_1$ de oscilación hasta el eje X deseado mediante la matriz de orientación $A^{x,x1}$ indicada en la ecuación 5 ·

$$A^{x,x1} = \begin{pmatrix} \cos(a) & 0 & -\sin(a) \\ 0 & 1 & 0 \\ \sin(a) & 0 & \cos(a) \end{pmatrix} \tag{5}$$

En la figura 10 se observa el desalineamiento entre los ejes principales de inercia y los ejes en torno a los cuales se está produciendo la oscilación en este último experimento. Procediendo según se indica con las ecuaciones siguientes se obtiene:

$$\overline{\overline{I}} = A^{x,x1} \cdot \overline{\overline{I_1}} \cdot \left(A^{x,x1}\right)^T \tag{6}$$

$$I_{xx} = I_{xx1} \cdot \left(1 - \sin^2(a)\right) + I_{zz1} \cdot \sin^2(a) \tag{7}$$

$$I_{zz} = I_{zz1} \cdot \left(1 - \sin^2(a)\right) + I_{xx1} \cdot \sin^2(a) \tag{8}$$

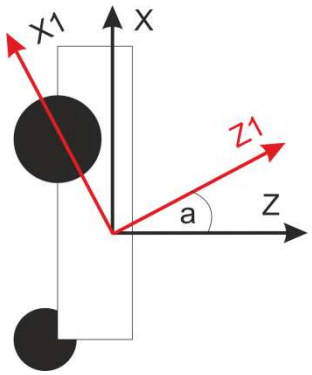



**Figura 10. Desalineamiento entre los ejes principales de inercia y los de oscilación durante el experimento**

Si a las ecuaciones (3), (4), (7) y (8), añadimos la ecuación (9) llegamos a un sistema determinado de 5 ecuaciones con 5 incógnitas del cual obtenemos los momentos principales de inercia buscados. Se ha tenido en cuenta que los ejes $yy$ e $yy1$ se mantienen paralelos.

$$I_{x1}^{P} = I_{zz1} + I_{yy1} = I_{zz1} + I_{yy} \tag{9}$$

Los resultados obtenidos de resolver el sistema de ecuaciones anterior resultan ser $I_{xx}$ = 0.0024, $I_{yy}$ = 0.00083 e $I_{zz}$ = 0.0021. Lo cual viene a corroborar muy aproximadamente lo obtenido del modelo CAD.

## 2.2 Modelo cinemático

Para llevar a cabo la confección del modelo cinemático se debe comenzar por realizar una modelización del vehículo real. De esta manera se pasa desde un sistema real complejo a una simplificación teórica que puede ser modelada computacionalmente. En la figura 11 se muestra el conjunto de sólidos original y su versión simplificada.

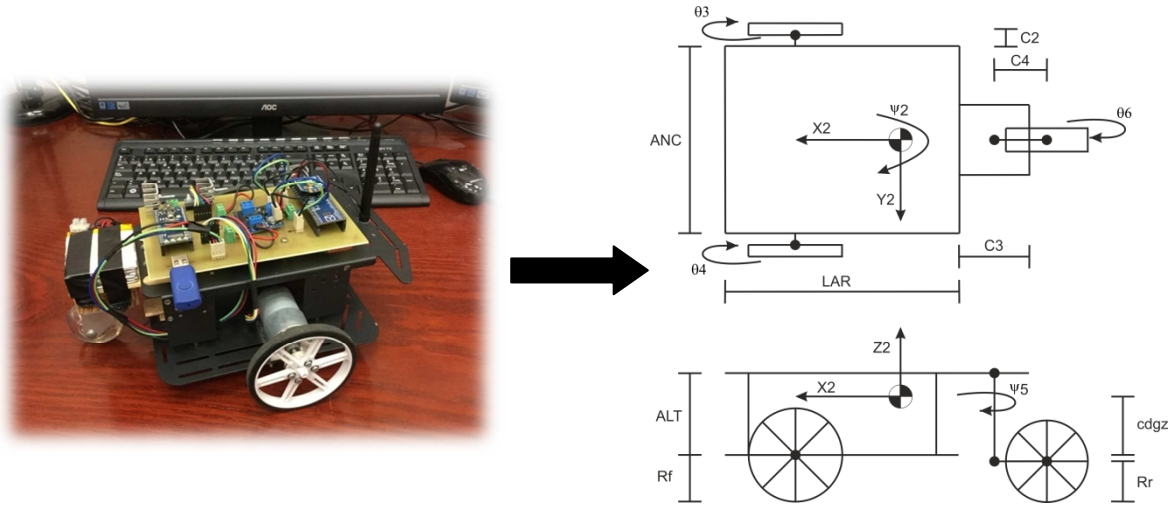



**Figura 11. Modelización del vehículo.**

A continuación es preciso tomar un conjunto de coordenadas que describirán la cinemática y la dinámica del sistema multicuerpo. En este caso se ha seleccionado un conjunto de siete coordenadas mostrado en la ecuación 10.

$$q = [x_2 \quad y_2 \quad \psi_2 \quad \theta_3 \quad \theta_4 \quad \psi_5 \quad \theta_6] \tag{10}$$

Para la generación del modelo supondremos un sistema multicuerpo con *nb* sólidos rígidos conectados entre sí mediante uniones cinemáticas. Cada cuerpo tendrá un sistema de referencia asociado <$G^i$, $x^i$, $y^i$, $z^i$> ubicado en el centro de gravedad del mismo. Denominaremos *n* al número de coordenadas utilizadas para describir el movimiento del sistema de sólidos respecto a un sistema de referencia global <*O, X, Y, Z*>.

La posición de un punto cualquiera del sistema multicuerpo será expresada como $r^i = r^i(q)$ y $A^i = A^i(q)$ es la matriz de orientación que transforma un vector de coordenadas expresado en el sistema de referencia de un cuerpo *i* al sistema de referencia global. Además existirá un conjunto de parámetros geométricos *p* característicos del sistema multicuerpo. De esta manera, $r^i = r^i(q, p)$ y $A^i = A^i(q, p)$.

También habrá un conjunto de *m* ecuaciones de restricción que condicionarán el movimiento del sistema. Estas restricciones pueden ser de distinto tipo en función de que factores se ven involucrados en ellas. Si sólo aparecen los valores de las coordenadas hablaremos de restricciones holónomas. Si además aparecen sus derivadas, hablaremos de restricciones de tipo no-holónomas. En función de que aparezca en ellas o no el tiempo tenemos restricciones

rehónomas o no rehónomas. El número de grados de libertad del sistema será $g = (n-m)$.

$$C(q, \dot{q}, t, p) = 0 \tag{11}$$

Las matrices de orientación de cada uno de los sólidos que integran el Róver visto como un sistema multicuerpo son:

$$A^2 = \begin{pmatrix} \cos(\psi_2) & -\sin(\psi_2) & 0 \\ \sin(\psi_2) & \cos(\psi_2) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{12}$$

$$A^{2,3} = \begin{pmatrix} \cos(\theta_3) & 0 & \sin(\theta_3) \\ 0 & 1 & 0 \\ -\sin(\theta_3) & 0 & \cos(\theta_3) \end{pmatrix} \tag{13}$$

$$A^{2,4} = \begin{pmatrix} \cos(\theta_4) & 0 & \sin(\theta_4) \\ 0 & 1 & 0 \\ -\sin(\theta_4) & 0 & \cos(\theta_4) \end{pmatrix} \tag{14}$$

$$A^{2,5} = \begin{pmatrix} \cos(\psi_5) & -\sin(\psi_5) & 0 \\ \sin(\psi_5) & \cos(\psi_5) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{15}$$

$$A^{5,6} = \begin{pmatrix} \cos(\theta_6) & 0 & \sin(\theta_6) \\ 0 & 1 & 0 \\ -\sin(\theta_6) & 0 & \cos(\theta_6) \end{pmatrix} \tag{16}$$

Se tiene que:

$$A^3 = A^2 A^{2,3} \tag{16}$$

$$A^4 = A^2 A^{2,4} \tag{17}$$

$$A^6 = A^2 A^{2,5} A^{5,6} \tag{18}$$

Las posiciones de los centros geométricos de los distintos sólidos vendrán dadas por las expresiones siguientes. En ellas $R_i$ es un vector expresado en el sistema de referencia global y $\overline{u}_i^j$ es la posición de un punto i del sólido j expresado en locales.

$$R_2 = \begin{bmatrix} x_2 & y_2 & \left(R_f + cdgz\right) \end{bmatrix}^T \quad (19)$$

$$R_3 = R_2 + A^2 \cdot \overline{u}_3^2 \quad (20)$$

$$R_4 = R_2 + A^2 \cdot \overline{u}_4^2 \quad (21)$$

Donde,

$$\overline{u}_3^2 = \begin{bmatrix} \left(LAR - C1 - cdgx\right) & -\left(\frac{ANC}{2} + C2\right) & -cdgz \end{bmatrix}^T \quad (22)$$

$$\overline{u}_4^2 = \begin{bmatrix} \left(LAR - C1 - cdgx\right) & \left(\frac{ANC}{2} + C2\right) & -cdgz \end{bmatrix}^T \quad (23)$$

Procediendo de forma análoga pueden definirse las posiciones de los sólidos 5, 6 y de los puntos de contacto de las ruedas con el suelo.

$$R_5 = R_2 + A^2 \cdot \overline{u}_5^2 \quad (24)$$

$$R_6 = R_5 + A^2 \cdot \overline{u}_6^5 \quad (25)$$

Donde,

$$\overline{u}_5^2 = \begin{bmatrix} -\left(cdg_x + \frac{C3}{2}\right) & 0 & \left(ALT - cdg_z\right) \end{bmatrix}^T \quad (26)$$

$$\overline{u}_6^5 = \begin{bmatrix} -C4 & 0 & -\left(ALT + R_F - R_R\right) \end{bmatrix}^T \quad (27)$$

De igual forma pueden expresarse los puntos de contacto de las ruedas con el suelo. En este caso, como veremos más adelante, interesa dejar expresadas dichas posiciones en el sistema de referencia local de cada rueda.

$$\overline{u}_P^3 = R_f \begin{bmatrix} sen\left(\theta_3\right) & 0 & -\cos\left(\theta_3\right) \end{bmatrix}^T \quad (28)$$

$$\overline{u}_Q^4 = R_f \begin{bmatrix} sen\left(\theta_4\right) & 0 & -\cos\left(\theta_4\right) \end{bmatrix}^T \quad (29)$$

$$\overline{u}_S^4 = R_f \begin{bmatrix} sen\left(\theta_6\right) & 0 & -\cos\left(\theta_6\right) \end{bmatrix}^T \quad (30)$$

Una vez se determinadas todas las posiciones se determinan las velocidades angulares y lineales de cada uno de los sólidos que forman el vehículo. Las velocidades angulares vendrán dadas por la ecuaciones 31 a 35 donde se utiliza la representación del vector a través de su matriz antisimétrica.

$$\widetilde{\overline{\omega_2}} = A_2{}^T \dot{A}_2 \tag{31}$$

$$\widetilde{\overline{\omega_3}} = A_3{}^T \dot{A}_3 \tag{32}$$

$$\widetilde{\overline{\omega_4}} = A_4{}^T \dot{A}_4 \tag{33}$$

$$\widetilde{\overline{\omega_5}} = A_5{}^T \dot{A}_5 \tag{34}$$

$$\widetilde{\overline{\omega_6}} = A_6{}^T \dot{A}_6 \tag{35}$$

Donde la formulación antisimétrica viene dada por la expresión 36.

$$\widetilde{\overline{\omega_i}} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{36}$$

Calculadas las velocidades angulares, se determinan a partir de ellas y de las posiciones de los sólidos las matrices de translaciones rotacionales. A partir de ellas pueden calcularse las velocidades lineales.

$$H_i = \frac{\partial R_i}{\partial q} \tag{37}$$

$$h_i = \frac{\partial \left( H_i \cdot \dot{q} \right)}{\partial q} \tag{38}$$

$$G_i = \frac{\partial \overline{\omega_i}}{\partial \dot{q}} \tag{39}$$

$$g_i = \frac{\partial \overline{\omega_i}}{\partial q} \tag{40}$$

La velocidad del centro de masas de cualquiera de los sólidos vendrá dada por la ecuación (41).

$$V_i = H_i \cdot \dot{q} \tag{40}$$

Una vez conocidas las velocidades se establecen las restricciones del movimiento. Para el caso que nos ocupa supondremos rodadura sin deslizamiento en las ruedas del vehículo. La velocidad de los puntos de contacto de las ruedas vendrá dada por las expresiones 41 a 43. Dada la construcción del vehículo, las dos ruedas delanteras se mantienen siempre paralelas al chasis, así pues la velocidad tangencial del punto de contacto será igual en ambas pero con signo opuesto. Así pues se impondrá que las

velocidades lineales de avance de las ruedas delanteras, y las velocidades lineal y tangencial de la trasera sean nulas.

$$V_P = V_3 + A_3 \cdot \left( \overline{\omega_3} \wedge \overline{u_P^3} \right) \tag{41}$$

$$V_Q = V_4 + A_4 \cdot \left( \overline{\omega_4} \wedge \overline{u_Q^4} \right) \tag{42}$$

$$V_S = V_6 + A_6 \cdot \left( \overline{\omega_6} \wedge \overline{u_S^6} \right) \tag{43}$$

Donde se han definido P,Q y S como los puntos de contacto con el suelo de las ruedas derecha, izquierda y trasera del vehículo respectivamente. Teniendo en cuenta lo enunciado en el párrafo anterior, las restricciones no-holónomas del movimientos vendrán dadas por las ecuaciones de la expresión 44.

$$\text{Rest}_{\text{Rod}} = \begin{bmatrix} V_{Px} \\ V_{Py} \\ V_{Qx} \\ V_{Sx} \\ V_{Sy} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{43}$$

Como se verá más adelante, para escribir las ecuaciones de la cinemática y la dinámica se necesitarán las expresiones del jacobiano de las restricciones y su derivada. Ambas expresiones están dadas por:

$$B_{Rod} = \frac{\partial \, \text{Re}\, s_{Rod}}{\partial \dot{q}} \tag{44}$$

$$\dot{B}_{Rod} = \frac{\partial \left( B_{Rod} \cdot \dot{q} \right)}{\partial q} \tag{45}$$

Las expresiones 10 a 45 constituyen el modelo cinemático del vehículo. Todas ellas se han introducido en Matlab simbólicamente y se han generado una serie de funciones que serán utilizadas para la simulación cinemática y simulación dinámica inversa. La ventaja de realizar los cálculos del modelo simbólicamente está en el hecho de que de esta manera se reduce el gasto computacional a la hora de realizar las simulaciones del modelo, ya que las expresiones están ya todas calculadas y sólo es necesario evaluarlas.

## 2.3 Modelo dinámico.

Al igual que con el modelo cinemático, el modelo dinámico se ha confeccionado simbólicamente en Matlab. Se han generado un serie de funciones simbólicas que posteriormente serán evaluadas durante la simulación dinámica inversa.

Las matrices de masas de inercias de cada uno de los sólidos que componen el vehículo serán expresados según las matrices de la expresión 46 y estarán particularizados para cada sólido según los datos indicados más arriba en este trabajo.

$$M_i = \begin{bmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \end{bmatrix} \qquad \bar{\bar{I}}_i = \begin{bmatrix} Ixx & Ixy & Ixz \\ Iyx & Iyy & Iyz \\ Izx & Izy & Izz \end{bmatrix} \tag{46}$$

Ensamblando las distintas matrices de masas e inercias se llega a la matriz de la expresión 47.

$$MM = \begin{bmatrix} M_2 & 0 & 0 & & & \\ 0 & \dots & 0 & & 0 & \\ 0 & 0 & M_6 & & & \\ & & & I_2 & 0 & 0 \\ & 0 & & 0 & \dots & 0 \\ & & & 0 & 0 & I_6 \end{bmatrix} \tag{47}$$

Las fuerzas generalizadas cuadráticas en velocidad vendrán dadas por:

$$QQ_v = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -\bar{\omega}_2 \wedge \left( \bar{\bar{I}}_2 \cdot \bar{\omega}_2 \right) \\ \dots \\ -\bar{\omega}_6 \wedge \left( \bar{\bar{I}}_6 \cdot \bar{\omega}_6 \right) \end{bmatrix} \tag{48}$$

Las fuerzas gravitatorias:

$$QQ_{grav} = \begin{bmatrix} 0 \\ 0 \\ -m_2 \cdot g \\ \dots \\ 0 \\ 0 \\ -m_6 \cdot g \\ 0 \\ \dots \\ 0 \end{bmatrix} \tag{49}$$

Y las fuerzas motrices:

$$Q_{mot} = G_3{}^T \cdot \begin{bmatrix} 0 \\ ParM3 \\ 0 \end{bmatrix} + G_4{}^T \cdot \begin{bmatrix} 0 \\ ParM4 \\ 0 \end{bmatrix} \tag{50}$$

Las matrices de transformación cinemáticas calculadas anteriormente se ensamblan de la siguiente forma:

$$L = \begin{bmatrix} H_2{}^T & \dots & H_6{}^T & G_2{}^T & \dots & G_6{}^T \end{bmatrix} \tag{51}$$

$$l = \begin{bmatrix} h_2{}^T & \dots & h_6{}^T & g_2{}^T & \dots & g_6{}^T \end{bmatrix} \tag{52}$$

Combinando las expresiones anteriores llegamos a la expresión de las ecuaciones del movimiento que en el próximo epígrafe serán manipuladas y ensambladas.

$$\begin{aligned} M &= L^T MML^T \\ Q_v &= L^T \left( QQv - MM \cdot l \cdot \dot{q} \right) \\ Q_{grav} &= L^T QQ_{grav} \end{aligned} \tag{53}$$

## 2.4 Simulación cinemática y dinámica inversa en Matlab del modelo empleando datos experimentales.

En el anexo 4 de este trabajo aparecen los códigos de Matlab programados para llevar a cabo la simulación cinemática. A continuación se describe brevemente cómo funciona el algoritmo y se muestran los resultados obtenidos de la simulación cinemática.

Lo primero que se hace es cargar los parámetros del modelos, es decir: masas, inercias y dimensiones que previamente han sido determinadas. A continuación se cargan los datos obtenidos en un experimento realizado con el vehículo. Dichos datos han sido previamente pre-procesados para introducirlos en un formato adecuado al programa de simulación. Los inputs al modelo serán exclusivamente los giros de ambas ruedas captados por los encoder y sus derivadas primera y segunda. Los demás datos de aceleraciones y velocidades angulares se usan exclusivamente para la validación del modelo. Decir que el experimento que se ha incluido en este trabajo consiste en un viaje del vehículo donde describe una trayectoria en L de ida y vuelta.

El siguiente paso es hacer una división de las coordenadas en dependientes e independientes. Teniendo en cuenta que el número total de coordenadas del problema es según (10) $n=7$ y que el número de ecuaciones de restricción según (43) es $m=5$, el número de grado de libertad del sistema será $g=n-m=2$. Esto tiene sentido ya que recordemos que las entradas físicas a nuestro sistema son las señales de control a los dos motores del vehículo. Teniendo en cuenta todo lo anterior tomaremos la siguiente división de coordenadas.

$$\begin{aligned} q_{ind} &= [\theta_3 \quad \theta_4] \\ q_{dep} &= [x_2 \quad y_2 \quad \psi_2 \quad \psi_5 \quad \theta_6] \end{aligned} \tag{54}$$

La simulación cinemática consistirá en ir resolviendo el modelo en cada instante de tiempo e ir determinando las coordenadas dependientes a partir de los valores de las independientes según las ecuaciones siguientes.

$$\dot{q}_{dep} = -\left(B_{Rod_{dep}}\right)^{-1}\left(B_{Rod_{ind}} \cdot \dot{q}_{ind}\right) \tag{55}$$

$$\ddot{q}_{dep} = -\left(B_{Rod_{dep}}\right)^{-1}\left(B_{Rod_{ind}} \cdot \ddot{q}_{ind} + \dot{B}_{Rod} \cdot \dot{q}\right) \tag{56}$$

Una vez conocidas las velocidades y aceleraciones independientes aplicando una simple regla de integración obtenemos los valores de las coordenadas dependientes.

$$x_2 = x_{2_0} + \dot{x}_2 \cdot \Delta t + \frac{1}{2} \cdot \Delta t^2 \cdot \ddot{x}_2 \quad (57)$$

$$y_2 = y_{2_0} + \dot{y}_2 \cdot \Delta t + \frac{1}{2} \cdot \Delta t^2 \cdot \ddot{y}_2 \quad (58)$$

$$\psi_2 = \psi_{2_0} + \dot{\psi}_2 \cdot \Delta t + \frac{1}{2} \cdot \Delta t^2 \cdot \ddot{\psi}_2 \quad (59)$$

$$\psi_5 = \psi_{5_0} + \dot{\psi}_5 \cdot \Delta t + \frac{1}{2} \cdot \Delta t^2 \cdot \ddot{\psi}_5 \quad (60)$$

$$\theta_6 = \theta_{6_0} + \dot{\theta}_6 \cdot \Delta t + \frac{1}{2} \cdot \Delta t^2 \cdot \ddot{\theta}_6 \quad (61)$$

La figura siguiente representa la trayectoria seguida por el vehículo, como puede observarse no se trata de una L perfecta de ida y vuelta. Ello se debe a que el modelo presenta el denominado *drift* y que los resultados obtenidos difieren de la realidad. El *drift* aparece debido a que al tener un modelo con restricciones de tipo no-holónomo no se puede ir resolviendo la posición (es decir los valores de las coordenadas) en cada iteración, sino que se están resolviendo unas restricciones función de las derivadas de las coordenadas. Ello implica que se vayan acumulando errores en cada iteración.

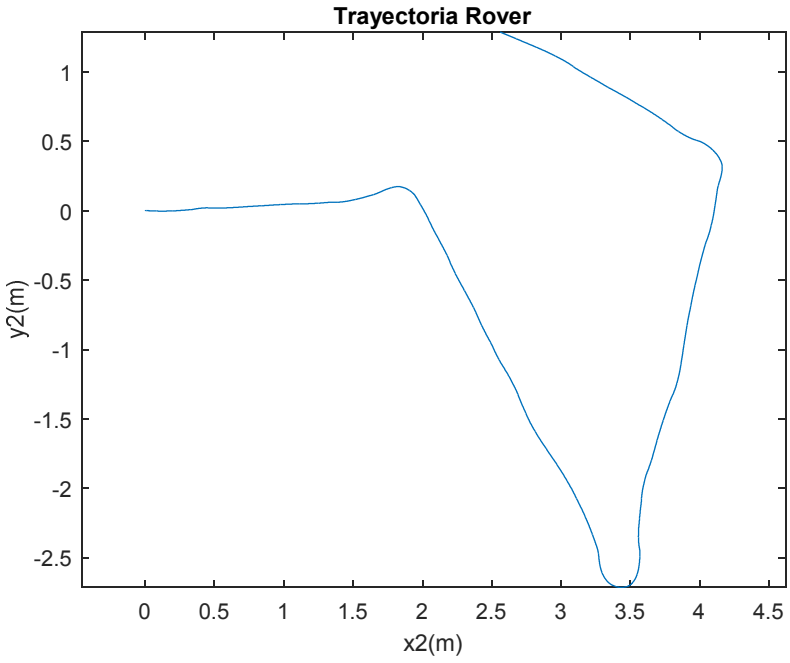



**Figura 12. Simulación cinemática de la trayectoria seguida por el vehículo**

Una vez calculados los valores de las coordenadas se puede validar el resultado de la simulación comparando las aceleraciones que experimentaría una IMU virtual situada en el mismo lugar del vehículo donde se haya colocado el sensor

físico. Para ello es preciso realizar un sencillo post-proceso a los datos obtenidos.

La velocidad angular experimentada por el cdg del chasis del vehículo vendrá dada por la ecuación (62). En la figura 14 se observa como la señal de velocidad angular en el eje z medida por la IMU y la simulada con el modelo casan casi a la perfección.

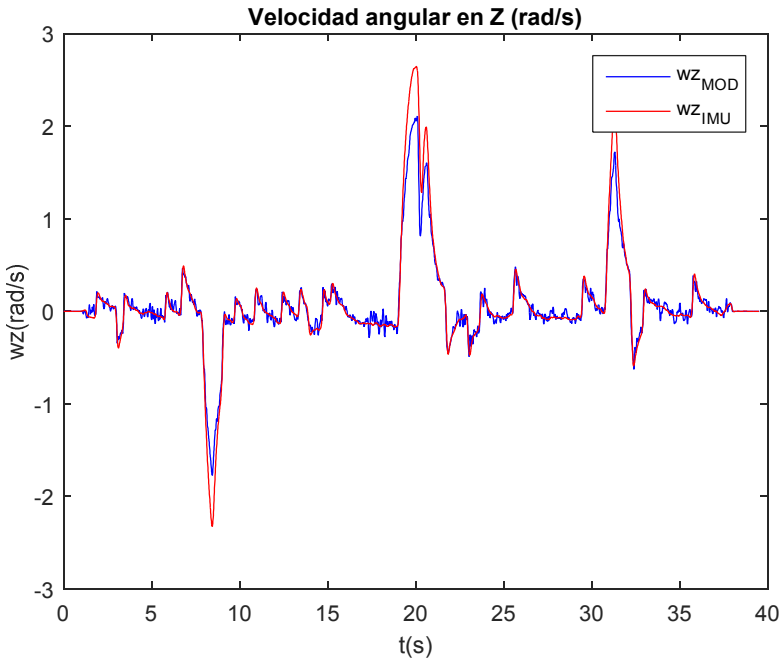



**Figura 13. Velocidades angulares en z medidas y simuladas**

Para hacer una comparación similar con las aceleraciones medidas y simuladas es preciso hacer una serie de manipulaciones previas. El proceso seguido es el descrito en las ecuaciones de la (62) a la (65). Donde $\overline{u}_{IMU}$ es la posición de colocación de la IMU expresada en el sistema de referencia del chasis.

$$\ddot{\overline{R}}^{IMU} = A_2^{\,T} \cdot \left( \ddot{R}_2 + \ddot{A}_2 \cdot \overline{u}_{IMU} \right) \tag{62}$$

$$\ddot{A}_2 = \left( \tilde{\alpha} + \tilde{\omega} \cdot \tilde{\omega} \right) \cdot A_2 \tag{63}$$

$$\overline{\alpha} = \overline{G}\ddot{q} + \overline{g}\dot{q} \tag{64}$$

$$a_{IMU} = \ddot{\overline{R}}^{IMU} + A_2^{\,T} \cdot \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{65}$$

En las figuras 15 y 16 se observa la comparación entre la aceleración lineal en $x$ e $y$ medidas por la IMU y generadas por el modelo. Cómo puede observase, la

aceleración en $y$ es calculada razonablemente bien, se aprecian perfectamente los giros efectuados por el vehículo para seguir la trayectoria en L. No ocurre lo mismo con la aceleración lineal en $x$ donde aunque siendo ambas señales iguales en magnitud no se capta totalmente su forma a partir de la simulación.

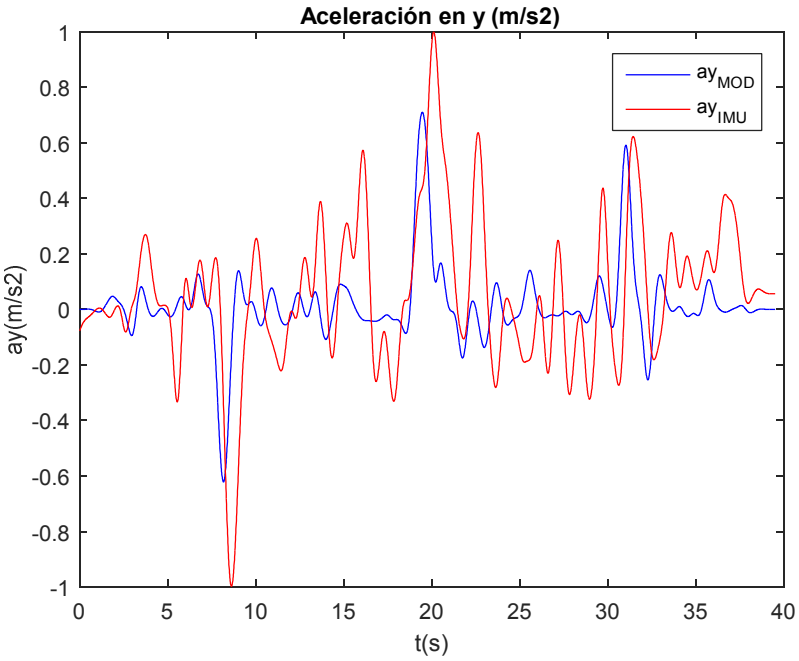



**Figura 14. Aceleraciones lineales en y simuladas y medidas**

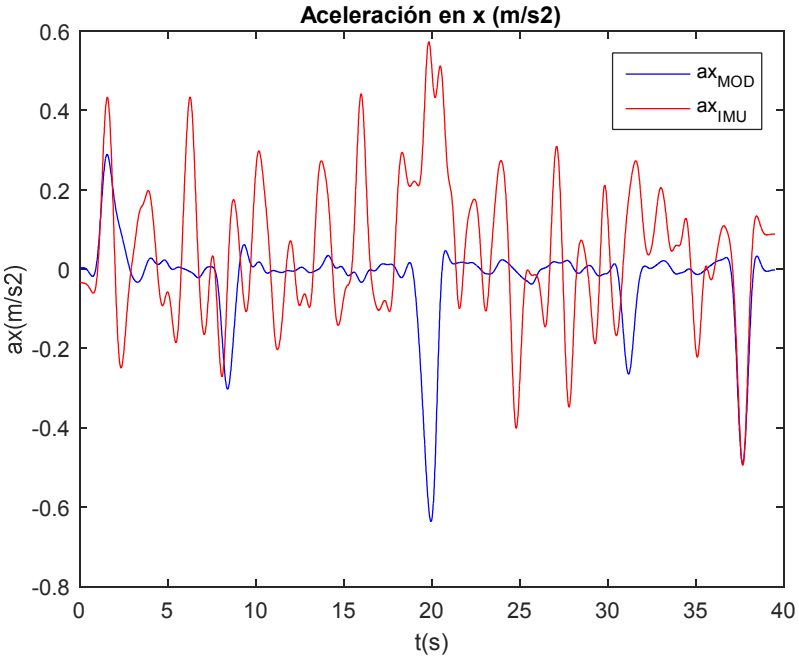



**Figura 15. Aceleraciones lineales en x simuladas y medias**

Para realizar la simulación dinámica inversa procedemos de forma similar. De dicha simulación obtendremos las fuerzas que hacen posible el movimiento a partir de los datos de entrada procedentes de los encoder. Al igual que en la simulación cinemática se mantiene la división en coordenadas dependientes e independientes indicada en (54). Las velocidades dependientes se obtiene a partir de las velocidades independientes a partir de la ecuación (55). Las aceleraciones se obtendrán mediante la resolución de las ecuaciones de Newton-Euler expresadas en (66) y (67). Teniendo en cuenta que del experimento conocemos los valores de aceleración de las coordenadas independientes obtenidos mediante derivación, dichos valores deben ser introducidos en las ecuaciones de Newton-Euler como inputs para a partir de ellos obtener el valor de las fuerzas motrices que ponen al vehículo en movimiento. Dichas fuerzas serán los pares motores aplicados a cada una de las ruedas.

Las ecuaciones de Newton-Euler vienen dadas por:

$$M\ddot{q} + B^T\lambda = Q_{ap} + Q_v \tag{65}$$

$$B\ddot{q} + \dot{B}\dot{q} = 0 \tag{66}$$

Separando los términos dependientes de los independientes llegamos a:

$$M_{dep}\ddot{q}_{dep} + M_{ind}\ddot{q}_{ind}^{\cdot\cdot} + B^T\lambda = Q_{M_3} + Q_{M_4} + Q_{grav} + Q_v \tag{67}$$

$$B_{dep}\ddot{q}_{dep} + B_{ind}\ddot{q}_{ind} + \dot{B}\dot{q} = 0 \tag{68}$$

Despejando términos conocidos a la derecha e incógnitas a la izquierda se llega a:

$$\begin{bmatrix} M_{dep} & -S & B^T \\ B_{dep} & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_{dep} \\ T \\ \lambda \end{bmatrix} = \begin{bmatrix} -M_{ind} \cdot q_{ind} + Q_{grav} + Q_v \\ -\dot{B}\dot{q} - B_{ind}\ddot{q}_{ind} \end{bmatrix} \tag{69}$$

Donde:

$$T = \begin{bmatrix} Par_{M_3} \\ Par_{M_4} \end{bmatrix} \tag{70}$$

$$S = \begin{bmatrix} \bar{G}_3^{\,T}(2,:) & \bar{G}_4^{\,T}(2,:) \end{bmatrix} \tag{71}$$

Resolviendo el sistema de ecuaciones de (69) se obtienen las aceleraciones dependientes. Conocidas ya las velocidades y aceleraciones dependientes, emulando el proceso de integración seguido en (57) a (61) se obtienen los valores de las coordenadas dependientes. En las gráficas siguientes se muestran la trayectoria simulada seguida por el vehículo y los pares motores en cada rueda. Como puede observarse la trayectoria sigue presentando el mismo drift que con la simulación cinemática.

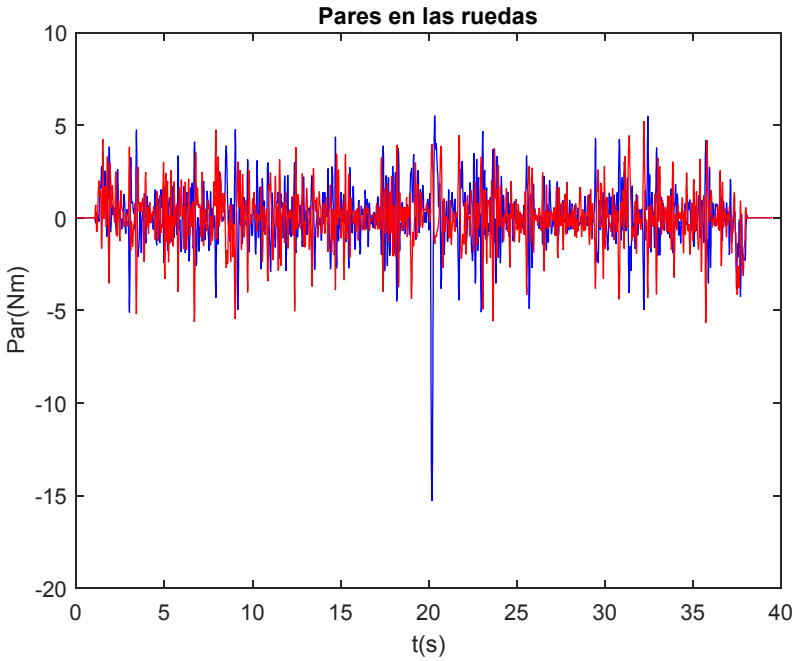



**Figura 16. Pares motores en las ruedas**

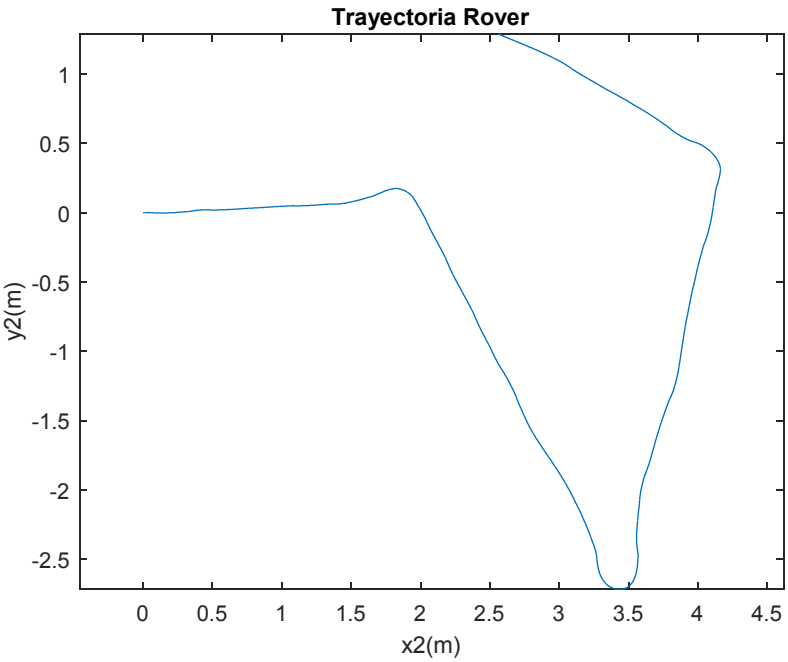



**Figura 17. Trayectoria integrada en la simulación dinámica inversa**

# Capítulo 3

# Sistema de visión computacional para la reconstrucción de la trayectoria

En este capítulo se describe el proceso de creación de un sistema de visión computacional para la reconstrucción de la trayectoria seguida por el vehículo. Se ha desarrollado un algoritmo capaz de identificar una trayectoria bidimensional a partir de las imágenes captadas por una video cámara durante el movimiento del vehículo. Finalmente se explican los detalles de la calibración y puesta a punto del sistema de visión artificial.

## 3.1 Reconstrucción de una trayectoria 2D mediante visión artificial

Se pretende reconstruir la trayectoria descrita por un punto P que se desplaza sobre el plano definido por los puntos G1, G2, G3 y G4 a partir de las imágenes tomadas por una cámara web ubicada en una posición arbitraria del espacio.

Supondremos que el punto P, cuya trayectoria queremos reconstruir, se moverá siempre sobre el plano de trabajo, cuyo sistema de referencia vendrá dado por el triedro < X, Y, Z >. De aquí en adelante lo denominaremos sistema de referencia global.

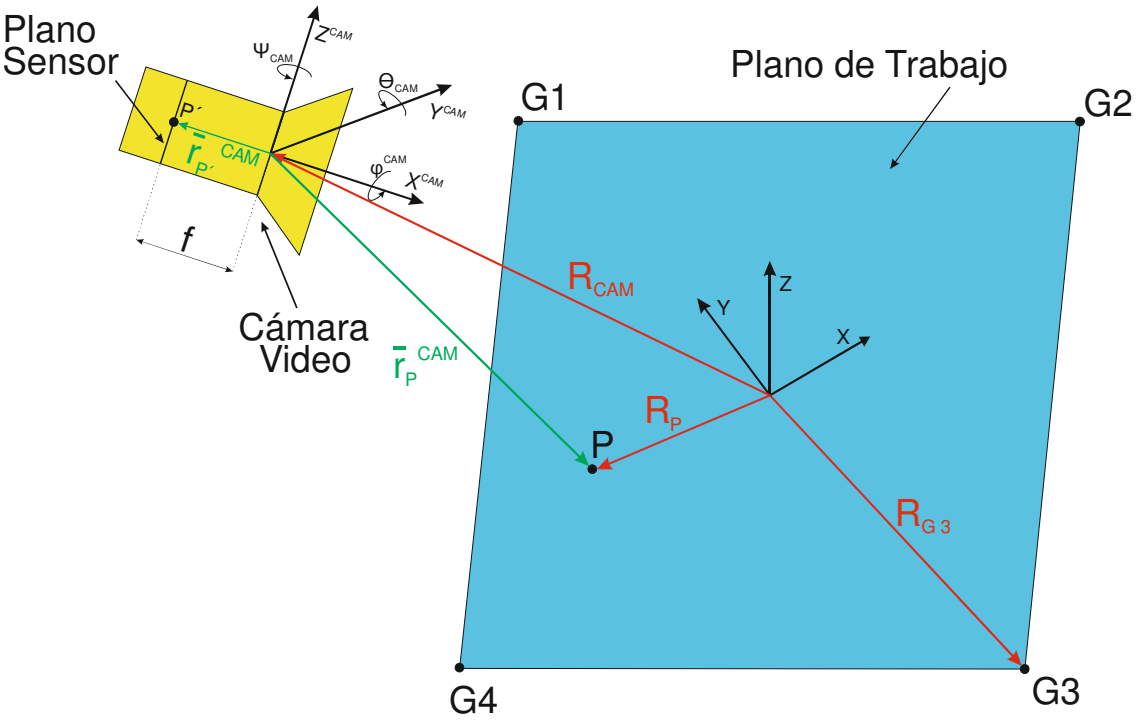



**Figura 18. Esquema de la cinemática del problema**

Las coordenadas de los puntos G1, G2, G3 y G4 respecto al sistema de referencia global se suponen conocidas y vendrán dadas por las expresiones (1) a (4).

$$R_{G1} = \begin{bmatrix} X_{G1} & Y_{G1} & 0 \end{bmatrix}^T \quad (72)$$

$$R_{G2} = \begin{bmatrix} X_{G2} & Y_{G2} & 0 \end{bmatrix}^T \quad (73)$$

$$R_{G3} = \begin{bmatrix} X_{G3} & Y_{G3} & 0 \end{bmatrix}^T \quad (74)$$

$$R_{G4} = \begin{bmatrix} X_{G4} & Y_{G4} & 0 \end{bmatrix}^T \quad (75)$$

Por otra parte, supondremos que la cámara de vídeo encargada de ir tomando los distintos fotogramas de la trayectoria, estará ubicada en una posición arbitraria y desconocida del espacio, dada por el vector de la expresión (76).

$$R_{CAM} = \begin{bmatrix} X_{CAM} & Y_{CAM} & Z_{CAM} \end{bmatrix}^T \quad (76)$$

Al mismo tiempo, el sistema de referencia de la cámara dado por el triedro $\langle X^{CAM}, Y^{CAM}, Z^{CAM} \rangle$ tendrá una determinada orientación respecto del sistema de referencia global, y estará dada por los ángulos de Euler $\varphi_{CAM}$, $\theta_{CAM}$ y $\psi_{CAM}$. La matriz de orientación del sistema de referencia de la cámara respecto del sistema de referencia global vendrá dada por la siguiente expresión.

$$A^{CAM} = A_{\psi_{CAM}} \cdot A_{\varphi_{CAM}} \cdot A_{\theta_{CAM}} = \begin{bmatrix} \cos(\psi_{CAM}) & -sen(\psi_{CAM}) & 0 \\ sen(\psi_{CAM}) & \cos(\psi_{CAM}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi_{CAM}) & -sen(\varphi_{CAM}) \\ 0 & sen(\varphi_{CAM}) & \cos(\varphi_{CAM}) \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta_{CAM}) & 0 & sen(\theta_{CAM}) \\ 0 & 1 & 0 \\ -\mathrm{s}\,en(\theta_{CAM}) & 0 & \cos(\theta_{CAM}) \end{bmatrix} \quad (77)$$

De forma general, la posición del punto P vendrá dada por la expresión 78.

$$R_P = R_{CAM} + A^{CAM} \cdot \overline{r}_P^{CAM} \quad (78)$$

### 3.1.1 Cinemática de la Visión Artificial

La figura 19 muestra el problema que aparece cuando hay que calcular la posición de objetos a partir de la imagen grabada por una cámara de video. En la figura se asume que la cámara se comporta como una cámara oscura con distancia focal f entre la lente y el plano sensor.

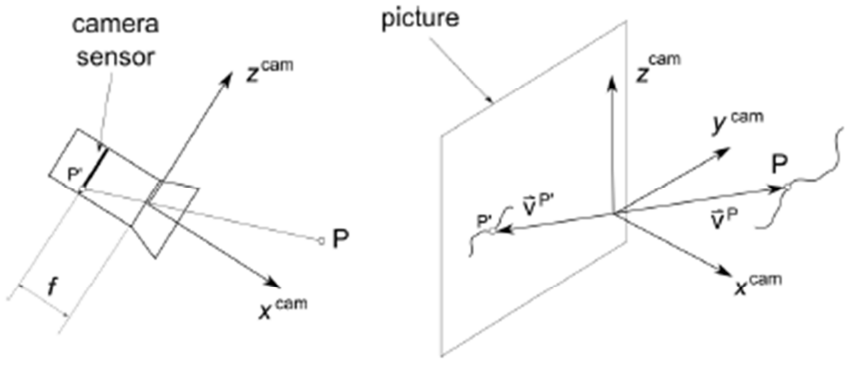



**Figura 19. Cinemática de la visión artificial**

La posición $\vec{v}^{P}$ de un punto P en la realidad con respecto al sistema de referencia de la cámara <$X_{CAM}$, $Y_{CAM}$, $Z_{CAM}$> puede relacionarse con la posición $\vec{v}^{P'}$ de la imagen captada por el plano sensor mediante la siguiente expresión.

$$\frac{\left(v^P\right)_x}{f} = \frac{\left(v^P\right)_y}{\left(v^{P'}\right)_y} = \frac{\left(v^P\right)_z}{\left(v^{P'}\right)_z} \quad (79)$$

Donde $\vec{v}^{P} = \left[(v^{P})_x \quad (v^{P})_y \quad (v^{P})_z\right]^T$ y $\vec{v}^{P'} = \left[f \quad \left(v^{P'}\right)_y \quad \left(v^{P'}\right)_z\right]^T$ son las componentes de los vectores de posición de los puntos P, en la realidad, y P´, en la imagen grabada, respectivamente. La ecuación (79) se deduce del modelo de cámara oscura de la cámara y los triángulos semejantes que forman las componentes de los vectores $\vec{v}^{P}$ y $\vec{v}^{P'}$. La ecuación (79) se puede reescribir de la forma:

$$\frac{f}{\left(v^{P}\right)_x} v^{P} = v^{P'} \tag{80}$$

Teniendo en cuenta que las componentes z e y de $\vec{v}^{P'}$ son conocidas a partir de la imagen grabada y supuestos conocidos los parámetros intrínsecos de la cámara (distancia focal f y el tamaño de los píxeles del sensor), de la ecuación (80) se pueden obtener 2 ecuaciones independientes para el cálculo de las tres componentes de $\vec{v}^{P}$. Como es de esperar, a partir de una imagen bidimensional no es posible obtener la tercera dimensión que falta en el problema. Para la reconstrucción de una trayectoria tridimensional sería preciso emplear una segunda cámara. Esto no alteraría el planteamiento realizado hasta entonces, simplemente aumenta el número de variables y ecuaciones que intervienen en el problema.

### 3.1.2 Simplificación para el caso 2D

Como se explica en el apartado anterior no resulta posible reconstruir una posición tridimensional a partir de la información captada por una sola cámara de video. Sin embargo, si lo que queremos es obtener la trayectoria descrita por un punto que se mueve sobre un único plano, es suficiente con un solo aparato.

Volviendo a la figura 18, supondremos conocidas las coordenadas de los puntos G1, G2, G3 y G4, dadas por las expresiones (72) a (75). De forma general la posición del punto G1 puede escribirse de la siguiente forma:

$$R_{G1} = R_{CAM} + A^{CAM} \cdot \bar{r}_{G1}^{CAM} \tag{81}$$

Donde:

$$\bar{r}_{G1}^{CAM} = \left[x_{G1} \quad y_{G1} \quad z_{G1}\right]^T \tag{82}$$

Si volvemos a relacionar la posición real del punto G1 captada por la cámara con la posición del mismo punto, sobre el plano sensor, G1´ en este caso, a partir de la ecuación (80) se llega a la siguiente expresión:

$$\frac{x_{G1}}{f} = \frac{y_{G1}}{y_{G1'}} = \frac{z_{G1}}{z_{G1'}} \tag{83}$$

Donde se ha tenido en cuenta que las componentes del vector $\bar{r}_{G1'}^{CAM}$ en el sistema de referencia de la cámara vienen dadas por la expresión $\bar{r}_{G1'}^{CAM} = [f \quad y_{G1'} \quad z_{G1'}]$.

De la ecuación (83) podemos extraer 2 ecuaciones linealmente independientes cuyas expresiones serán:

$$x_{G1} \cdot y_{G1'} - f \cdot y_{G1} = 0 \tag{84}$$

$$z_{G1'} \cdot y_{G1} - z_{G1} \cdot y_{G1'} = 0 \tag{85}$$

Utilizando las dos ecuaciones anteriores, el vector de posición en la realidad del punto G1 visto desde el sistema de referencia de la cámara puede ser expresado de la siguiente manera:

$$\overset{-CAM}{r_{G1}} = \begin{bmatrix} x_{G1} \\ y_{G1} \\ z_{G1} \end{bmatrix} = \begin{bmatrix} \frac{f \cdot y_{G1}}{y_{G1'}} \\ y_{G1} \\ \frac{z_{G1'} \cdot y_{G1}}{y_{G1'}} \end{bmatrix} = y_{G1} \cdot \begin{bmatrix} \frac{f}{y_{G1'}} \\ 1 \\ \frac{z_{G1'}}{y_{G1'}} \end{bmatrix} \tag{86}$$

Sustituyendo (86), (76), (77) y (72) en la ecuación (81) llegamos a un sistema de tres ecuaciones con siete incógnitas.

$$\begin{bmatrix} X_{G1} \\ Y_{G1} \\ 0 \end{bmatrix} = \begin{bmatrix} X_{CAM} \\ Y_{CAM} \\ Z_{CAM} \end{bmatrix} + A^{CAM} \cdot \left( y_{G1} \cdot \begin{bmatrix} \frac{f}{y_{G1'}} \\ 1 \\ \frac{z_{G1'}}{y_{G1'}} \end{bmatrix} \right) \tag{87}$$

Si procedemos de igual forma conocidas las posiciones en el sistema de referencia global de los puntos G2, G3 y G4 llegamos a las siguientes ecuaciones.

$$\begin{bmatrix} X_{G2} \\ Y_{G2} \\ 0 \end{bmatrix} = \begin{bmatrix} X_{CAM} \\ Y_{CAM} \\ Z_{CAM} \end{bmatrix} + A^{CAM} \cdot \left( y_{G2} \cdot \begin{bmatrix} \frac{f}{y_{G2'}} \\ 1 \\ \frac{z_{G2'}}{y_{G2'}} \end{bmatrix} \right) \tag{88}$$

$$\begin{bmatrix} X_{G3} \\ Y_{G3} \\ 0 \end{bmatrix} = \begin{bmatrix} X_{CAM} \\ Y_{CAM} \\ Z_{CAM} \end{bmatrix} + A^{CAM} \cdot \left( y_{G3} \cdot \begin{bmatrix} \frac{f}{y_{G3'}} \\ 1 \\ \frac{z_{G3'}}{y_{G3'}} \end{bmatrix} \right) \tag{89}$$

$$\begin{bmatrix} X_{G4} \\ Y_{G4} \\ 0 \end{bmatrix} = \begin{bmatrix} X_{CAM} \\ Y_{CAM} \\ Z_{CAM} \end{bmatrix} + A^{CAM} \cdot \left( y_{G4} \cdot \begin{bmatrix} \frac{f}{y_{G4'}} \\ 1 \\ \frac{z_{G4'}}{y_{G4'}} \end{bmatrix} \right) \tag{90}$$

Como puede observarse, sólo con considerar tres puntos ya se llega a un sistema de nueve ecuaciones y nueve incógnitas. No obstante, teniendo en cuentas que se trata de ecuaciones no lineales y que van a ser resueltas computacionalmente mediante algoritmos numéricos podemos incluir puntos conocidos extras que ayudarán a la convergencia del método. Eso es lo que se ha hecho al añadir el punto G4 a sistema final de ecuaciones compuesto por las expresiones (87), (88), (89) y (90).

## 3.2 Calibración del sistema de visión computacional

Para llevar a cabo la reconstrucción de la trayectoria seguida por un punto arbitrario P que se desplaza sobre un plano conocido, partimos de la base de tener una videocámara ubicada en una determinada posición del espacio, filmando dicho plano. Según avanza el tiempo, la videocámara irá tomando instantáneas a una frecuencia determinada que irán almacenando información sobre la posición del punto P en cada instante.

Supongamos que tenemos un sistema de visión computacional como el que se muestra en la figura 20. En ella puede observarse una cámara web enfocando un plano cuadriculado. Las dimensiones del plano son conocidas, cada cuadrícula tiene unas dimensiones de 20mm x 20mm. A partir de este sistema podemos llevar a cabo la calibración de nuestro sistema de visión artificial,

entendida esta calibración como la determinación de la ubicación y orientación espacial de la cámara de video.

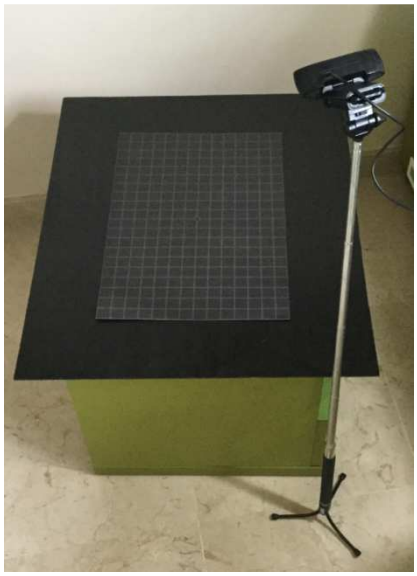


**Figura 20. Sistema de visión artificial.**

Si procesamos una imagen captada por la cámara desde Matlab, veremos como para el programa, no es más que una híper-matriz de tres capas, donde las otras dos dimensiones coinciden con la resolución de la imagen grabada. Cada una de estas capas estará llena de números comprendidos entre 0 y 255. Cada celda de esas capas representa un pixel de la imagen.

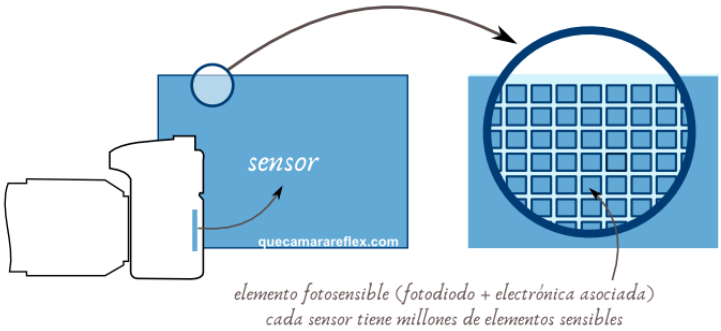



**Figura 21. Elemento fotosensible cámara de fotos.**

Como se observa en la figura 21, el plano sensor está formado por cientos de elementos fotosensibles. A mayor número de elementos mayor será la

resolución, mayor será el número de pixeles, que pueda tener la imagen captada por la cámara.

Cada elemento fotosensible tendrá una geometría plana concreta. Los píxeles pueden ser cuadrados o rectangulares. Lo más habitual es que sean cuadrados y tengan unas 5 micras por lado.

El tamaño del pixel del plano sensible y la distancia focal de la cámara son los denominados parámetros intrínsecos de la cámara. Conocido el tamaño del pixel, podemos transformar la posición expresada en pixeles de la imagen computarizada a coordenadas del sistema de referencia de la cámara (figura 19).

## 3.3 Puesta a punto del sistema de visión artificial

Una vez definido el problema cinemático de la visión computacional se ha realizado una serie de experimentos con el vehículo a control remoto. De estos experimentos se pretende extraer la trayectoria seguida por el vehículo cuando este rueda por una determinada área de control. En la figura 22 se muestra el montaje del sistema de visión artificial. Como puede observarse se ha ubicado la cámara en una posición tal que es capaz de captar totalmente el área captada en el suelo. Aprovechando las baldosas del piso y teniendo en cuenta que sus dimensiones son de 400 x 400 mm quedan definidas las coordenadas de las esquinas respecto al punto central.

Para llevar a cabo la calibración se toma una instantánea del área con la video cámara y se introduce en el algoritmo de calibración. En él se introducen manualmente las posiciones en píxeles de las esquinas del área de control junto con sus cotas reales. Ejecutando el algoritmo este devuelve la posición de la cámara respecto del centro del plano.

Para este experimento concreto la posición calculada por el programa resulta ser las indicadas en (91). Por medición directa en el plano se comprobó que las medidas eran efectivamente esas.

$$\begin{aligned} X_{CAM} &= -1.68m \\ Y_{CAM} &= -0.04m \\ Z_{CAM} &= 1.38m \end{aligned} \tag{91}$$

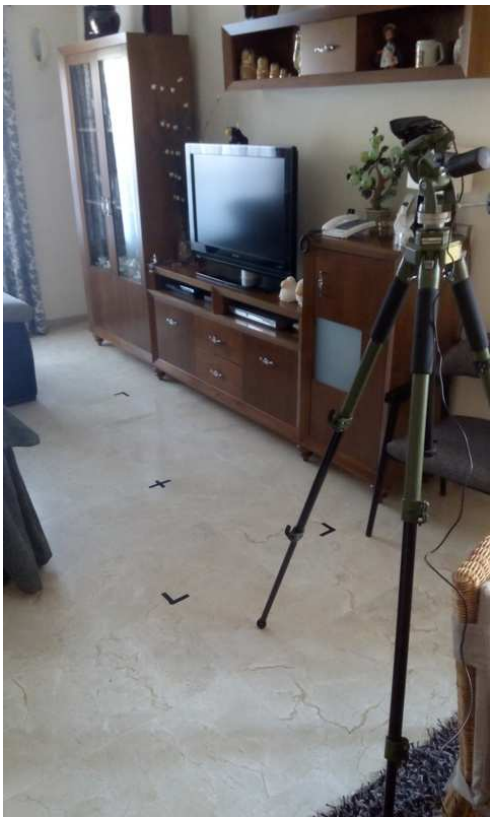


**Figura 22. Montaje del sistema de visión artificial**

Una vez calibrado el sistema se puede dar inicio a la experimentación. Para ello se hace rodar el vehículo dentro del área de control y se grava su movimiento con la video cámara. Existen múltiples algoritmos que permiten detectar el movimiento de un objeto en un conjunto de imágenes. En este trabajo no se ha utilizado ninguno de esos algoritmos sino que se ha hecho un seguimiento a un punto fijo del vehículo. Ese punto es un led blanco que va montado sobre el chasis del vehículo. Con un ajuste correcto de parámetros en la cámara, puede hacerse esta menos insensible a otras tonalidades y hacerla captar perfectamete la luz del diodo. En la figura 23 se observa un fotograma del video grabado durante el experimento. El algoritmo programado en Matlab va localizando y almacenando en cada iteración la posición del led blanco en cada fotograma.

**Figura 23. Fotograma captado durante el experimento de reconstrucción de trayectorias**

Una vez procesado el vídeo y obtenidas las posiciones en píxeles que el vehículo ha ido ocupando se introducen esos datos en el algoritmo de visión artificial. Tras resolver las ecuaciones (87) a (90) devolverá las posiciones reales en el plano por donde el vehículo a pasado. En la figura 24 se muestra la reconstrucción de la trayectoria realizada por el algoritmo.

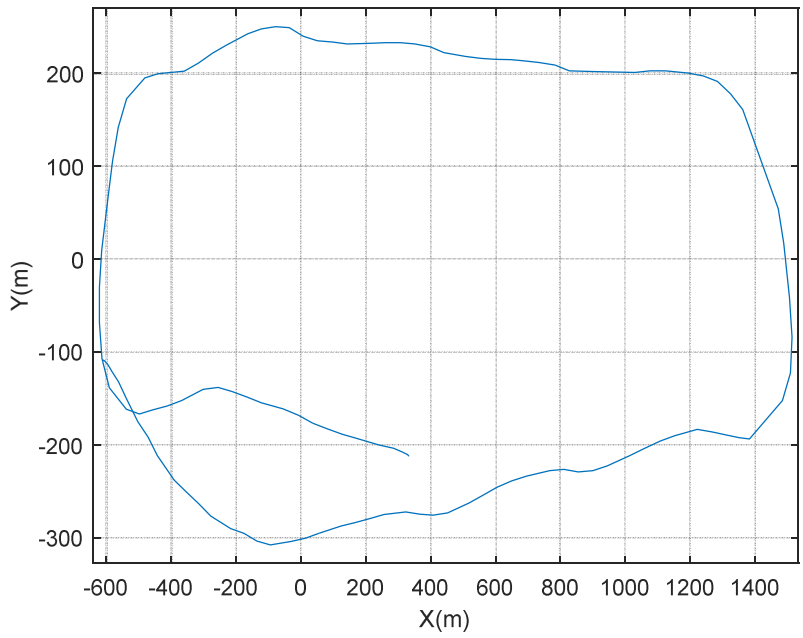



**Figura 24. Trayectoria reconstruida por el algoritmo de visión artificial**

Si tomamos los datos adquiridos por la IMU y los encoder de las ruedas y los introducimos en el modelo cinemático o dinámico utilizado en el capítulo anterior podemos ver la contrastación entre la trayectoria real seguida por el vehículo (a partir del algoritmo de visión artificial) y la simulada por el programa.

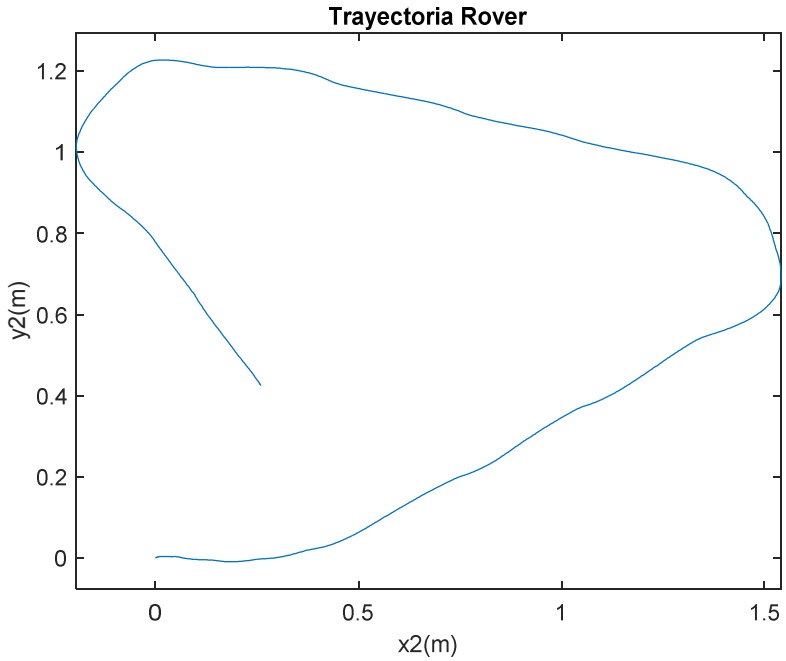



**Figura 25. Trayectoria simulada**

En la figura 26 se representan las velocidades angulares reales y simuladas, como puede observarse hay un alto grado de acuerdo entre ambas señales. La figura 25 muestra la trayectoria simulada y como puede observarse no llega a ser cerrada como la real. Ello se debe al igual que antes al drift presentado por el modelo.

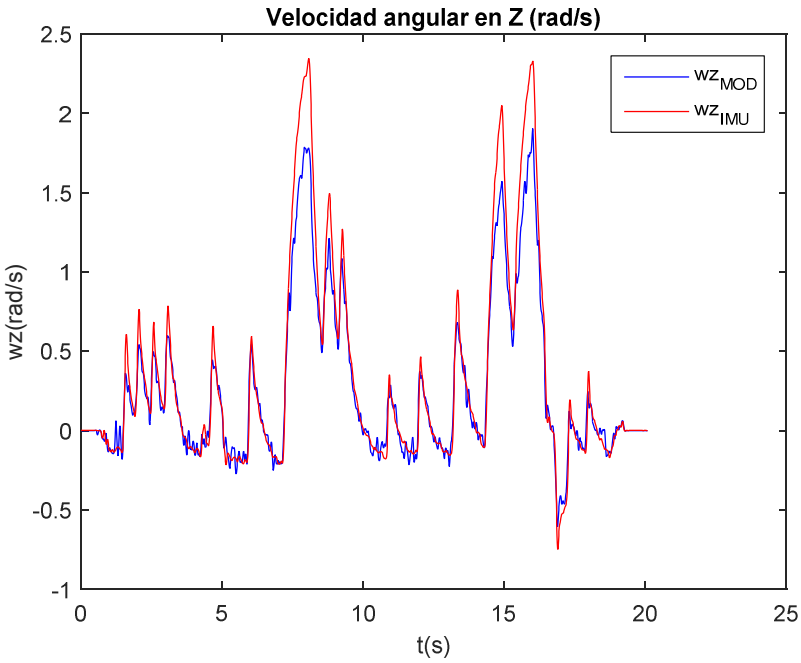



**Figura 26. Velocidades angulares reales y simuladas**

# Capítulo 4

# Electrónica del vehículo

En este segundo capítulo se explican los detalles del proceso de rediseño y fabricación de la circuitería del vehículo y del mando a distancia. También se recoge todo lo relativo a nuevo sistema de control del vehículo y al nuevo sistema de adquisición de datos. Finalmente se presentan algunos resultados obtenidos durante la puesta a punto del sistema.

## 4.1 Rediseño de la circuitería del vehículo y del mando a distancia

Para dar solución a todos los problemas electrónicos mencionados en el capítulo primero, se ha llevado a cabo un rediseño y construcción del hardware del vehículo y de su mando a distancia. En la figura 27 y tabla 3 se muestran e indican los distintos elementos que componen la placa base del vehículo.

Como puede observarse, las diferencias respecto al diseño original son notables. Uno de los principales avances tecnológicos introducidos ha sido pasar de una placa base soldada sobre una baquelita perforada estándar, a un circuito impreso diseñado y construido a medida para esta aplicación.

El diseño del circuito integrado principal se ha diseñado con Eagle®, software específico para el diseño de este tipo de aplicaciones electrónicas. En la figura 28 se muestra el diseño de pistas resultante de Eagle®. Una vez validado el diseño se procedió a su fabricación mediante el método convencional de

insolación, revelado y ataque con ácido. El resultado obtenido fue altamente satisfactorio.

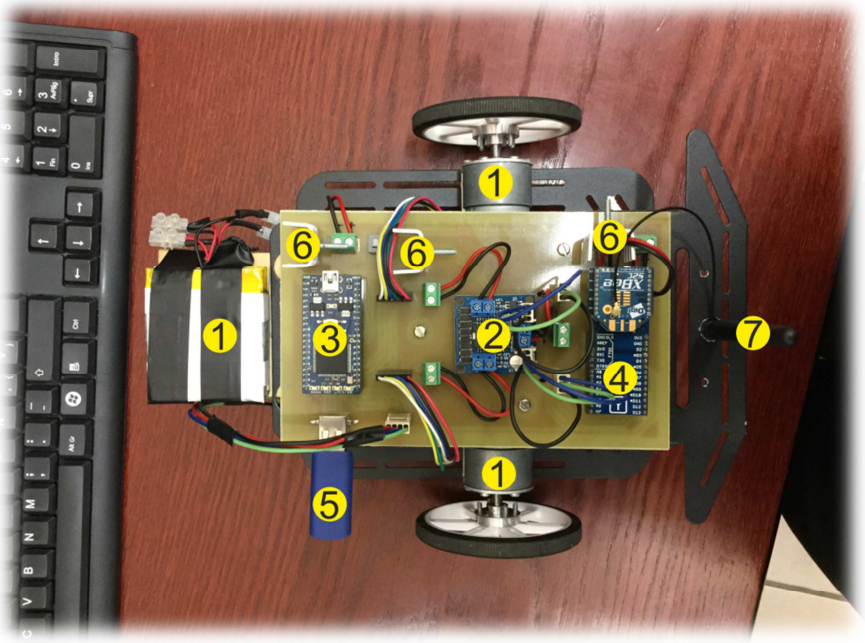



**Figura 27. Vista en planta del Róver**

Una vez fabricada la placa, se realizaron los taladros de los pines y se soldaron los componentes. Todos los componentes se han soldado a la placa y no son removibles con excepción de los microcontroladores y el puente H que están instalados sobre racks de pines soldados a la placa.

La alimentación de la placa se realiza a partir del paquete de tres baterías Lipo conectadas en serie, que suministran una tensión nominal de aproximadamente 12 V. Con este voltaje sólo puede ser alimentado directamente el puente H que lleva integrado su propio regulador de corriente. Para dar energía a los microcontroladores y los encoder se han instalado limitadores de tensión a 5 y 9 voltios del tipo LM78xx acompañados de disipador de calor.

**Tabla 2. Componentes electrónicos de la placa base del Róver**

| Número | Nombre del componente |
|---|---|
| **1** | Motor DC con encoder modeloPololu 30:1 Metal Gearmotor 37Dx52L mm 64 CPR Encoder. |
| **2** | Puente H modelo estándar con 2 salidas. |
| **3** | Microcontrolador Mbed LPC1768. |
| **4** | Microcontrolador Arduino Fio + Xbee RF. |
| **5** | Dispositivo de almacenamiento externo formato FAT32. |
| **6** | Limitador de tensión LM78xx con disipador de calor. |
| **7** | Antena receptora. |

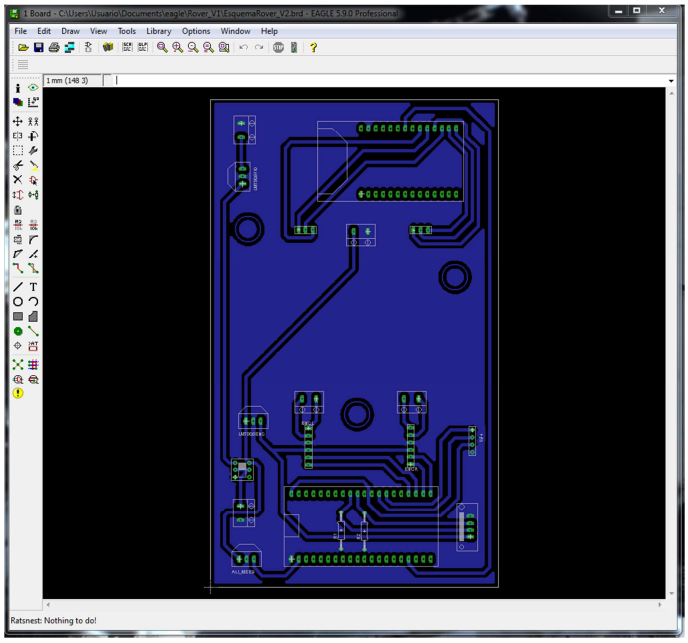



**Figura 28. Placa base del Róver en Eagle®**

Por último, la IMU utilizada en el vehículo es una unidad inercial compuesta de dos sensores. Un acelerómetro BMA180 y un giróscopo ITG3200. El dispositivo se conecta a través de protocolo $I^2C$ a la Mbed. Se ha instalado dentro del chasis del vehículo en una posición muy próxima al centro de masas del vehículo como puede observarse en la figura 29. La IMU, junto con los encoders y la Mbed constituyen el sistema de adquisición de datos del vehículo.

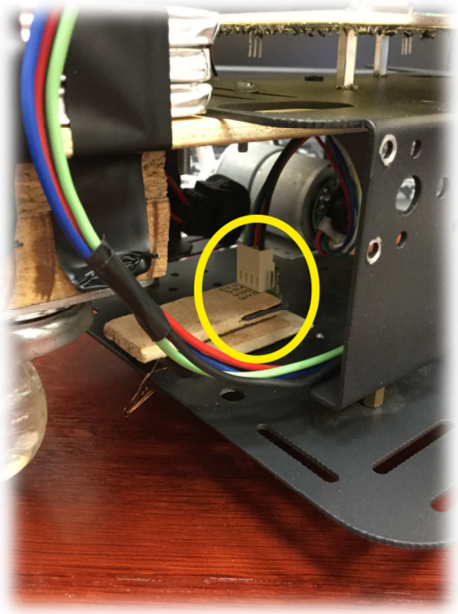


**Figura 29. Ubicación de la IMU en el vehículo**

De igual forma que para la placa base del vehículo se describen a continuación las características del mando a distancia y sus diferentes funciones. En la figura 30 y tabla 4 se muestran y describen los componentes del dispositivo.

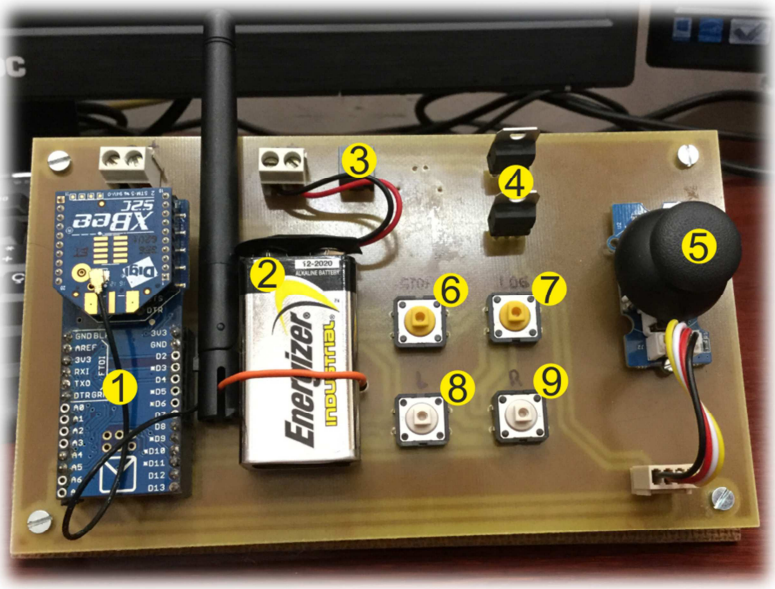



**Figura 30. Componentes del mando a distancia**

**Tabla 3. Componentes electrónicos del mando a distancia.**

| Número | Nombre del componente |
|---|---|
| **1** | Microcontrolador Arduino Fio + Xbee RF. |
| **2** | Batería 9V. |
| **3** | Interruptor On/Off. |
| **4** | Limitadores de tensión tipo LM7805sin disipador. |
| **5** | Joystick analógico 5V. |
| **6** | Botón STOP. |
| **7** | Botón LOG. |
| **8** | Botón corregir deriva a izquierda. |
| **9** | Botón corregir deriva a derecha. |

La tecnología de fabricación del mando a distancia ha sido idéntica a la de la placa base del vehículo. La alimentación se realiza a partir de una pila convencional de 9V cuya tensión es reducida y estabilizada a 5V por dos limitadores tipo LM7805 para suministrar energía al microcontrolador y al joystick.

La botonera está enrutada con el microcontrolador. Su misión es la de enviar señales discretas a una serie de pines del mismo y en función de su valor producir diferentes acciones de control. El botón STOP se utiliza para detener completamente los motores del vehículo, el botón LOG inicia y finaliza la adquisición de datos. Los botones R y L corrigen la deriva lateral del vehículo como se explicará en el siguiente epígrafe. El joystick como era de esperar, se utiliza para controlar los movimientos del vehículo.

En el anexo 1 se muestra el esquema eléctrico simplificado de la placa base del vehículo y del mando a distancia.

## 4.2 Algoritmo de control del vehículo

Como se ha comentado anteriormente, el control de los motores del vehículo se realiza por medio del controlador Arduino Fio. El algoritmo de control programado es un control en bucle abierto, es decir, no se retroalimenta de la información de los encoder para saber en cada instante la velocidad de cada motor.

El motivo de haber elegido esta forma de control se debe a que el microcontrolador Arduino no tiene suficiente potencia como para gestionar el cálculo de velocidad de ambos motores, la ejecución del algoritmo PID de control de la velocidad y el control remoto del vehículo simultáneamente.

Inicialmente se realizaron diversas pruebas pero el comportamiento del vehículo no fue en absoluto el esperado.

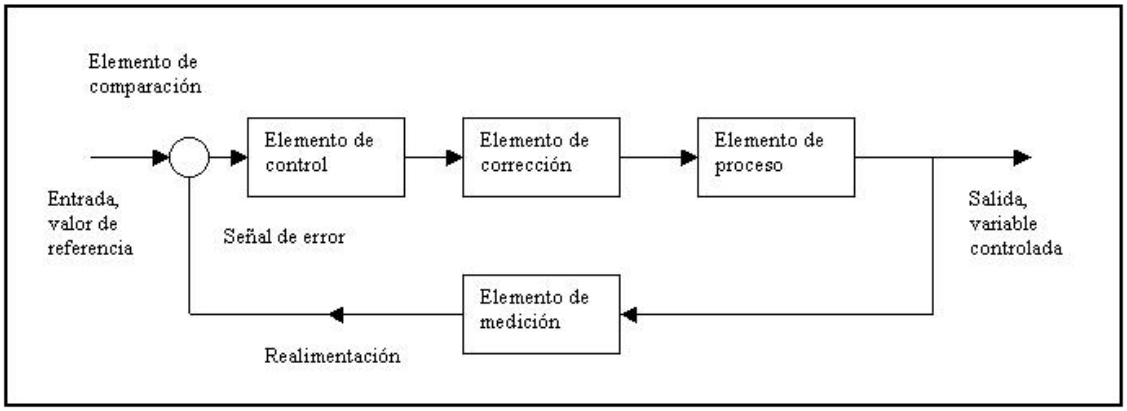



**Figura 31.Control lazo cerrado**

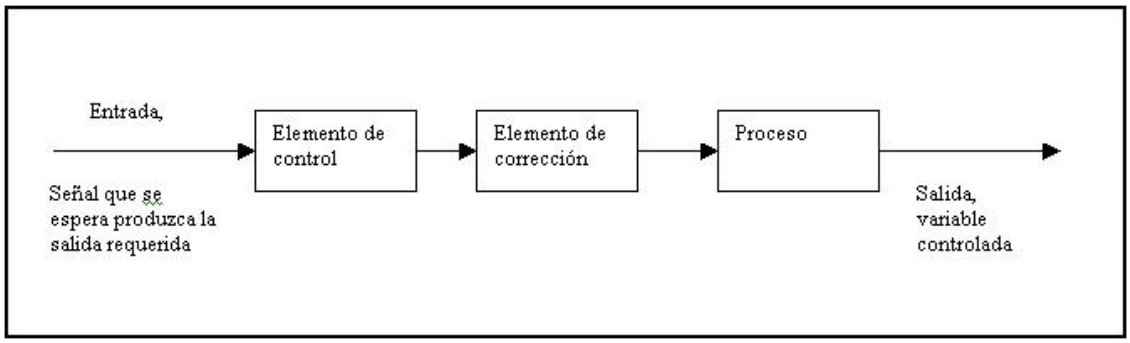



**Figura 32. Control lazo abierto**

Con el control en lazo abierto lo que se hace es enviar directamente a ambos motores una señal tipo PWM proporcional a la velocidad de avance que se quiere conseguir.

Supuesto que ambos motores son idénticos, si la señal de control es la misma en ambos, el vehículo debe avanzar en línea recta. La realidad es que ni los motores son exactamente iguales ni tampoco lo son las señales de control que salen del puente H. Como resultado aparece una leve discrepancia entre la velocidad de giro de ambos motores respecto de la consigna establecida. Para solucionar este problema se determina experimentalmente un factor que multiplicará la señal de control de uno de los motores de manera que la velocidad resultante de ambos sea la misma.

También ocurre que, conforme va cayendo la tensión de la batería, empiezan a aparecer nuevas discrepancias entre las velocidades de giro de las ruedas y el vehículo empieza a desviarse hacia un lado. Para corregir esta deriva están los botones R y L del mando, cuya función es la de ir modificando el valor de dicho factor de corrección. Un par de pulsaciones en uno de los sentidos suele ser suficiente para devolver al vehículo a la trayectoria deseada.

La programación del Arduino emisor del mando y del Arduino receptor del vehículo se ha hecho a través de la IDE de Arduino. Los detalles de la programación se recogen en el anexo 2.

Parte esencial del control es la comunicación inalámbrica entre el mando a distancia y el vehículo. Para llevarla a cabo se creado una red de comunicación mediante dos módulos de radio frecuencia Xbee. La configuración de la red se realiza a través de la XCTU de las Xbee. Software que puede descargarse gratuitamente desde la página web del fabricante. En dicha web se incluyen una serie de tutoriales que rápidamente permiten al usuario no experto poner a funcionar el sistema.

El control del movimiento del vehículo se realiza mediante el joystick del mando a distancia. El sistema funciona de la siguiente manera: Si se mantiene el joystick pulsado hacia delante el vehículo va ganando velocidad hasta su valor máximo. Si antes de llegar a alcanzar ese máximo se deja de pulsar el vehículo mantiene la velocidad actual. Al pulsar el joystick en sentido contrario, la velocidad de avance se va reduciendo, llegando cero si se mantiene presionado el tiempo suficiente.

Para hacer girar el vehículo lo que se hace es reducir ligeramente la velocidad del motor situado en el lado hacia donde se quiere girar. Si se mantiene pulsado el joystick hacia un la derecha o izquierda, el motor de ese lado irá reduciendo progresivamente su velocidad hasta llegar si se quiere a detenerse. De esta manera se puede conseguir que el vehículo pivote sobre esa rueda. Si lo que se quiere es variar sólo un poco la trayectoria con un leve toque lateral al joystick se consigue el efecto deseado.

## 4.3 Sistema de adquisición de datos

El elemento esencial de nuestro vehículo auscultador de superficies es su sistema de adquisición de datos. Dicho sistema está integrado por un microcontrolador Mbed, dos encoder, una IMU y un dispositivo de almacenamiento USB.

La programación del microcontrolador se ha hecho en C++ a través del compilador online de Mbed. En el anexo 3 se adjunta el código completo implementado.

El sistema funciona de  la siguiente manera: Se ha establecido una conexión entre el microcontrolador Arduino Fio del Róver y la Mbed, de manera que cuando el Arduino recibe del mando la señal de inicio o finalización de la adquisición este transfiere esta señal a la Mbed. En el esquema eléctrico de la placa base puede verse dicha conexión.

Una vez la Mbed recibe la señal de disparo, da comienzo el proceso de adquisición de datos. Como resultado, a la finalización del ensayo se generará en el dispositivo de almacenamiento USB un fichero .txt con los datos adquiridos. El fichero tendrá nueve columnas cuyos datos serán: tiempo (s), $a_x$ ($m/s^2$), $a_y$($m/s^2$), $a_z$($m/s^2$), $w_x$($^o/s$),$w_y$($^o/s$), $w_z$($^o/s$), pulsos rueda derecha y pulsos rueda izquierda.

En el USB se irá almacenando un fichero por cada adquisición que se realice, entendida esta como el periodo comprendido entre dos pulsaciones consecutivas del botón LOG del mando. Los ficheros tendrán siempre el nombre dat_Vxx.txt donde xx será un número de dos cifras comprendido entre el 00 y el 99 que irá incrementándose de uno en uno tras cada ensayo.

Para llevar la cuenta de los ensayos en la memoria de Mbed habrá siempre un fichero donde se almacene el número del último ensayo grabado. De esta forma al iniciarse el proceso de adquisición se primero dicho fichero y a partir del número que tiene almacenado se genera en el USB el fichero donde se almacenarán los datos.

La IMU utilizada consta de dos sensores embebidos en un mismo integrado. Se trata de un acelerómetro de 14 bits BMA180 y un giróscopo de 16 bits ITG3200, alimentados a 3,3V desde la Mbed. El acelerómetro viene por defecto programado para trabajar en la escala +/- 2g, rango que mantendremos intacto. Así pues, teóricamente el acelerómetro devolverá un número entre -8192 y 8192, valores correspondientes al fondo de escala de -2g y +2g respectivamente. Por su parte el ITG3200 trabaja en un rango cerrado de +/- 2000($^o/s$). Ambos sensores habrá que someterlos a un sencillo proceso de calibración que se explica en el epígrafe siguiente.

Por último, los encoder tienen una resolución de 960 pulsos por revolución según las especificaciones del fabricante. La salida del encoder son dos señales cuadradas de 5V desfasadas entre sí. La existencia de estos dos canales puede permitir bien determinar el sentido de giro del motor o aumentar la resolución

del mismo. Dado que para los experimentos que se van a realizar no se considera la posibilidad de que el vehículo avance marcha atrás se ha optado por emplear ambos canales en un solo sentido aumentando de esta manera la resolución del dispositivo.

## 4.4 Puesta a punto del sistema de adquisición

Una vez se ensamblado en el vehículo el sistema de adquisición de datos hay que proceder a calibrarlo.

La calibración de los encoders y del ITG3200 es prácticamente inmediata. Con respecto a los encoder sólo hay que tener en cuenta que, dado que los motores están instalados de forma simétrica en el vehículo en unos de los encoder los pulsos se irán incrementando conforme avanza el vehículo y en el otro se irán decrementando. Como resultado, en el fichero de datos de salida se observa que una de las columnas de los encoder tiene número positivos y la otra negativos. Esto se soluciona sencillamente pre-multiplicando por menos uno la columna negativa en el código en C++ de la Mbed.

La calibración del ITG3200 no resulta inmediata de realizar, por ello el fabricante ya nos da en el datasheet el factor por el que hay que multiplicar la salida del sensor para convertirla a la unidad de salida deseada. En este caso resulta ser (1/14,375) para pasar de bits a $^{o}$/s.

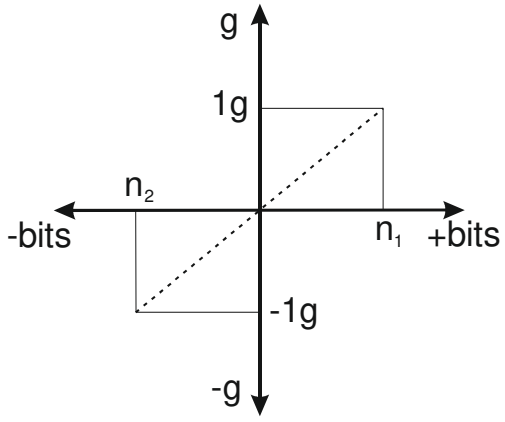



**Figura 33. Calibración del acelerómetro en cada eje**

El acelerómetro podría calibrarse directamente a partir de los datos del fabricante, aunque lo ideal es realizar una calibración propia. El proceso es sencillo y rápido de ejecutar: Una vez el sistema de adquisición está a punto se

deben realizar seis ensayos estáticos con el acelerómetro y adquirir datos. En cada ensayo se colocará el sensor en dirección a cada uno de sus ejes tanto en sentido positivo como negativo. Dado que el dispositivo está en reposo, la aceleración medida debe ser la gravedad. Con una simple regla de tres se llega al factor multiplicador para cada eje. Por ejemplo si colocamos el sensor paralelo al plano horizontal mediremos la gravedad en el eje +Z. Si le damos la vuelta estaremos también midiendo la gravedad pero en –Z en este caso. Con estos dos valores obtenemos la pendiente de la recta que coincidirá con dicho factor de calibración. Repitiendo la misma operación con los otros dos ejes se tiene el acelerómetro calibrado. Los valores obtenidos pueden verse en el anexo III.

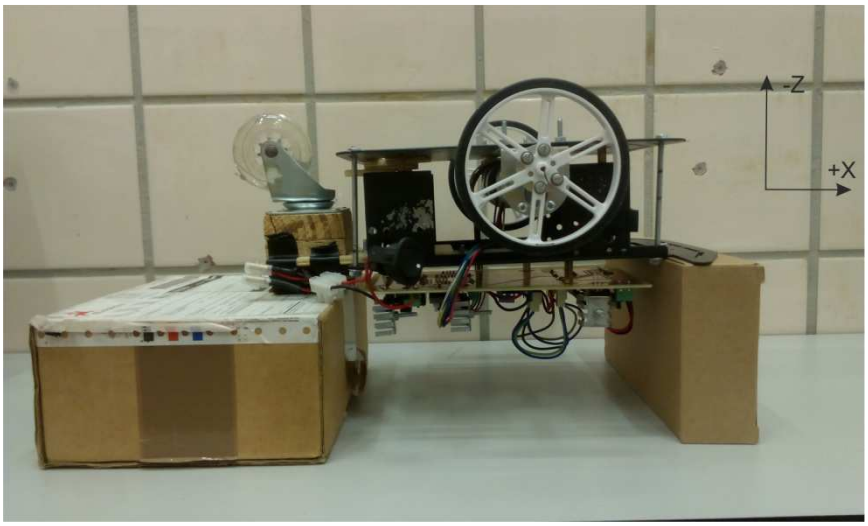



**Figura 34. Calibración del acelerómetro en el eje –Z**

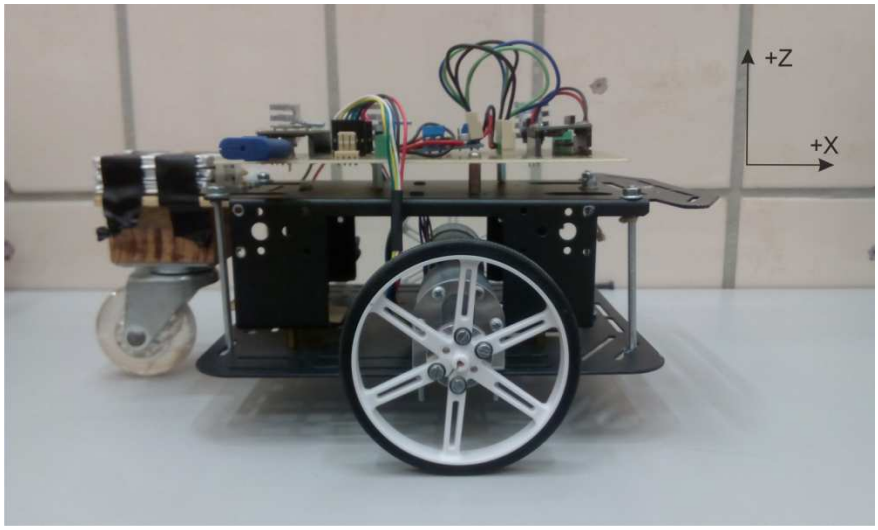



**Figura 35. Calibración del acelerómetro en el eje +Z**

# Capítulo 5

# Conclusiones y trabajos futuros

En este trabajo se ha llevado a cabo la remodelación electromecánica y de control de un vehículo a control remoto denominado *Róver*. El objetivo original de este trabajo era el de poner a punto dicho sistema para conseguir disponer de una herramienta sólida que pueda ser utilizada en distintas tareas investigadoras y académicas del Departamento de Ingeniería Mecánica de la Universidad de Sevilla.

Se ha logrado satisfactoriamente realizar una identificación de parámetros del vehículo así como la confección de un modelo cinemático y dinámico que ha demostrado funcionar adecuadamente para la aplicación requerida.

Complementariamente se ha diseñado y testeado a partir del vehículo y su modelo un sistema de visión computacional capaz de reconstruir la trayectoria seguida por el vehículo a partir de una grabación realizada durante el experimento.

Después de todo el trabajo realizado se han detectado algunos problemas que podrían ser solucionados en futuros trabajos con este sistema. Probablemente el principal de ellos sea al *drift* que aparece en las simulaciones debido a la utilización de restricciones de tipo no-holónomo en el modelo. Dado el tipo de sistema con el que estamos trabajando se podría modificar el modelo para

incluir otro tipo de restricción. Otra opción sería incluir un filtro de Kalman que combinase la información del modelo con el algoritmo de visión artificial. De esta manera se podría corregir el *drift* presente en el modelo.

Otra futura línea de trabajo con el vehículo es la utilización del filtro de Kalman para la reconstrucción del perfil del terreno por el que el vehículo va circulando. Este proceso resulta sumamente interesante y es extensible a otras aplicaciones de la industria como por ejemplo la ferroviaria, donde a partir de un modelo del vehículo y un sensor inercial instalado en el mismo se puede llevar a cabo la reconstrucción de la geometría vertical de la vía.

Con este proyecto se deja en el Departamento de Ingeniería Mecánica un sistema polivalente que podrá ser utilizado para labores tanto investigadoras como académicas.

# Bibliografía

[1] Apuntes de la asignatura Cinemática y Dinámica de Máquinas.

[2] Manual online Mbed: https://developer.mbed.org/handbook/Homepage

[3] Manual online Arduino: https://playground.arduino.cc/Es/Es

[4] J. García de Jalón, E. Bayo, Kinematic and Dynamic Simulation of Multibody Systems.

[5] A. Shabana, Dynamics of Multibody Systems.

[6] A. Shabana, E. Zaazaa, H. Sugiyama, RailRoad Vehicle Dynamics.

[7] D. Gross, W. Huger, J. Schröder, A. Wall, S. Govindjee, Engineering Mechanics 3.

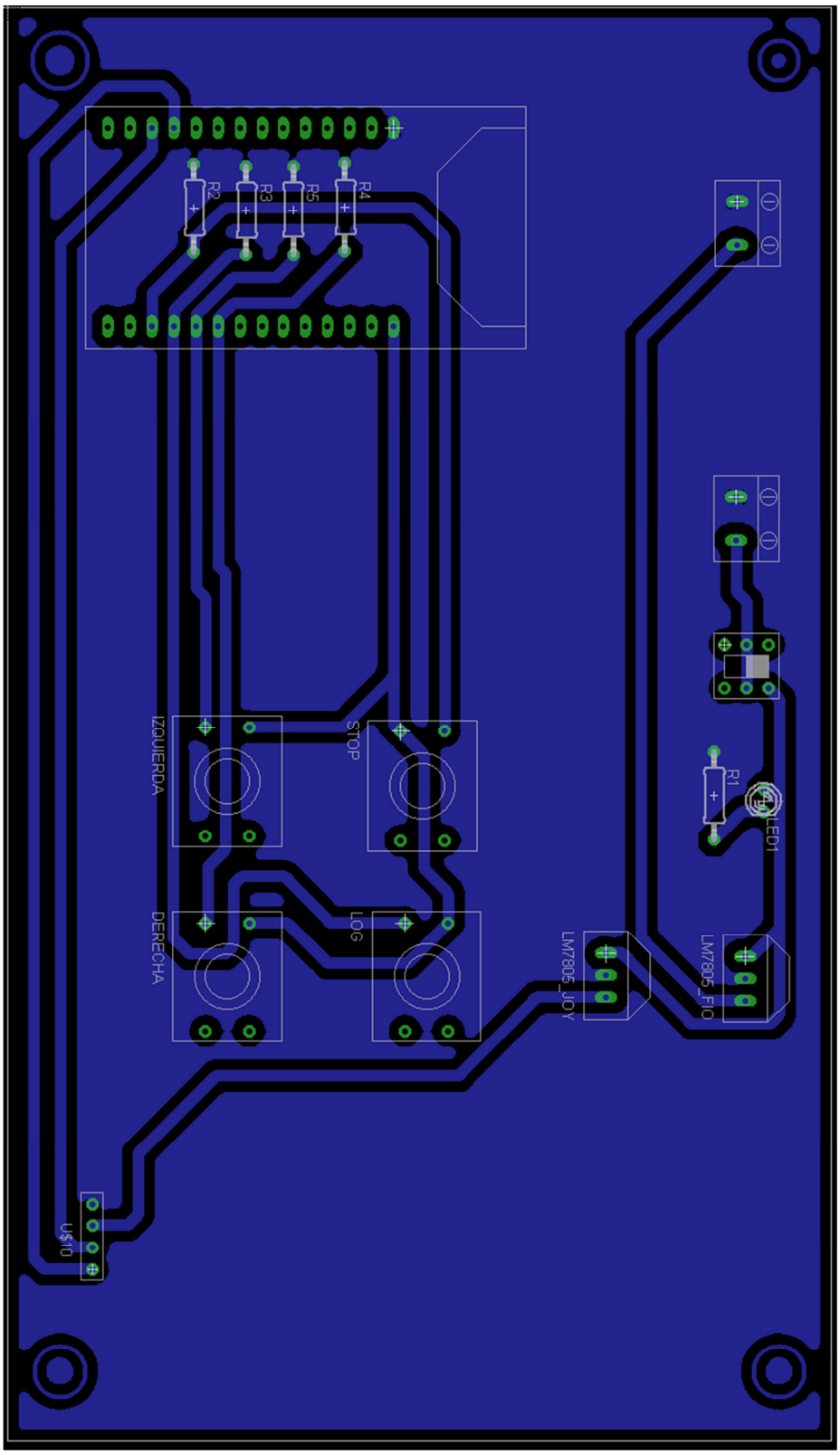
[8] Sen M. Kuo, Real-Time Digital Signal Processing.

[9] H. Press, A. Teukilshy, Numerical Recipes, The art of scientific computing.

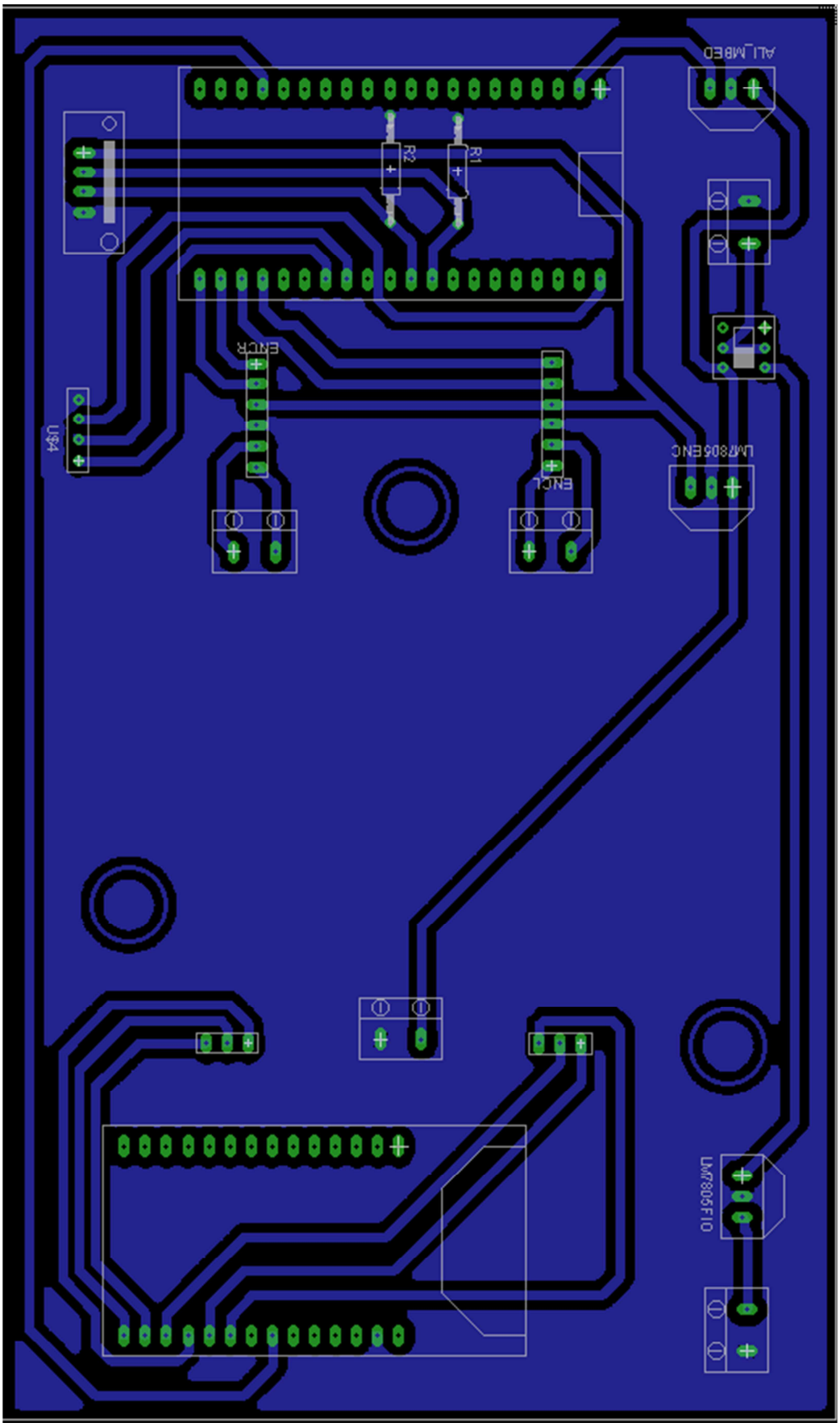
# Anexo 1

Esquemas eléctricos

## A1.1 Esquema eléctrico del mando



## A1.2 Esquema eléctrico de la placa base



# Anexo 2

# Algoritmo de control del vehículo

## A2.1 Algoritmo del mando a distancia

```
int analogPin4 = 4;
int analogPin5 = 5;

intParar = 11;
int Grabar = 10;
int Izquierda = 9;
int Derecha = 8;


int v4 = 0;
int v5 = 0;

void setup() {
Serial.begin(9600);
pinMode(Parar,INPUT);
pinMode(Grabar,INPUT);
pinMode(Izquierda,INPUT);
pinMode(Derecha,INPUT);
}

void loop() {
  v4 = analogRead(analogPin4);
  v5 = analogRead(analogPin5);

if ( v4 > 900 && (v5 > 600 && v5 < 900)){
Serial.println("A");
  }
if ( v4 < 600 && (v5 > 600 && v5 < 900)){
Serial.println("B");
  }
if ( (v4 > 600 && v4 < 900) && v5 < 600){
Serial.println("D");
```

```
  }
if ( (v4 > 600 && v4 < 900) && v5 > 900){
Serial.println("I");
  }
if( (v4 > 600 && v4 < 900) && (v5 > 600 && v5 < 900)){
Serial.println("C");
  }
if(digitalRead(Parar) == HIGH){
Serial.println("S");
  }
if(digitalRead(Grabar) == HIGH){
Serial.println("G");
}
if( digitalRead(Derecha) == HIGH){
Serial.println("R");
  }
if( digitalRead(Izquierda) == HIGH){
Serial.println("L");
  }
delay(10);
}
```

## A2.2 Algoritmo del vehículo

```
intVel;
float Vel1;
float Vel2;
floatGiroD;
floatGiroI;
int P1D = 8;
int P2D = 11;
int P1I = 12;
int P2I = 13;
int Mot1 = 9;
int Mot2 = 10;
intestado = 0;
char letter0 = 'X';
intmaximo = 300;
int logging = 6;

booleangrabar = false;

float factor = 0.87;
floatcorreccion = 0.003;

void setup() {
pinMode(P1D,OUTPUT);
pinMode(P2D,OUTPUT);
```

```
pinMode(P1I,OUTPUT);
pinMode(P2I,OUTPUT);

digitalWrite(P1D,HIGH);
digitalWrite(P2D,LOW);
digitalWrite(P1I,HIGH);
digitalWrite(P2I,LOW);

analogWrite(Mot1,0);
analogWrite(Mot2,0);

Serial.begin(9600);

pinMode(logging,OUTPUT);
digitalWrite(logging,LOW);
}

void loop() {
if(Serial.available()>0){
char letter = Serial.read();
if (letter == 'A'){
if (estado<maximo){
estado++;
       letter0 = letter;
     }
   }
if (letter == 'B'){
if (estado> 0){
estado--;
       letter0 = letter;
     }
   }
if (letter == 'C'){
     letter0 = letter;
GiroD = 1;
GiroI = 1;
   }
if (letter == 'I'){
GiroI = 0.4;
GiroD = 1.4;
     letter0 = letter;
   }
if (letter == 'D'){
GiroD = 0.4;
GiroI = 1.4;
     letter0 = letter;
   }
if (letter == 'S'){
estado = 0;
```

```
   }
if (letter == 'G'){
grabar = !grabar;
digitalWrite(logging,grabar);
   }
if (letter == 'R'){
factor = factor + correccion;
   }
if (letter == 'L'){
factor = factor - correccion;
}

Vel = map(estado,0,maximo,0,200);
   Vel1 = Vel*GiroD;
   Vel2 = Vel*GiroI;

 }
analogWrite(Mot1,Vel2);
analogWrite(Mot2,Vel1*factor);
 }
```

# Anexo 3

# Programación en C++ de la Mbed

## A3.1 Algoritmo del mando a distancia

```
#include "mbed.h"
#include "RingBuffer.h"
#include "BMA180_mod.h"
#include "ITG3200.h"
#include "QEI.h"
#include "USBHostMSD.h"

intboton = 0;
intbotonantes = 0;
intbotonahora = 0;
inttipoAdquisicion = 0;
float tiempo = 0;

intContadorEnsayos = 0;

int16_taX, aY, aZ;
intwX, wY, wZ;

float Acel3[3] = {0,0,0};
float Gyro3[3] = {0,0,0};

float a_xBias3, a_yBias3, a_zBias3;
int w_xBias3, w_yBias3, w_zBias3;

intpprev = 32;
intpulsos_R = 0, pulsos_L = 0;

QEI wheel_R (p21, p22, NC, pprev);
QEI wheel_L (p23, p24, NC, pprev);

ITG3200 gyro3(p28,p27);
BMA180 acc3(p28,p27);


/// PEN DRIVE ///
USBHostMSD *msc;
```

```
LocalFileSystemlocal("mbedFlash");

DigitalOutled(LED1);
DigitalOutled2(LED2);
DigitalOutled3(LED3);
DigitalOutled4(LED4);

DigitalInboton_Pin(p17);

Timer timer;

FILE * fp;
FILE * fp2;

voidadquisition(void);
voidGuardaDatosBuffer(void);
voidcalibracion(void);
voidinicializaIMU(void);

Ticker tickerLectura;

intsizeBuff = 100;
intPasoTiempo = 10000;
unsignedintContadorDatos = 0;

Buffer buff_aX(sizeBuff);
Buffer buff_aY(sizeBuff);
Buffer buff_aZ(sizeBuff);

Buffer buff_wX(sizeBuff);
Buffer buff_wY(sizeBuff);
Buffer buff_wZ(sizeBuff);

Buffer buff_wheel_R(sizeBuff);
Buffer buff_wheel_L(sizeBuff);

int main() {

msc = new USBHostMSD("local");

   led4 = 1;
if(!msc->connected()){
while(!msc->connect()) {
      }
   }
   led4 = 0;

inicializaIMU();
   led3=1;
calibracion();
   led3=0;

tickerLectura.attach_us(&adquisition,PasoTiempo);
timer.start();
led = 0;
```

```
charNoViaje[2];

while(1) {
boton = boton_Pin.read();
botonantes = botonahora;
botonahora = boton;

if (botonantes == 1 &&botonahora == 0) {

ContadorDatos = 0;

if (tipoAdquisicion == 0) {
tipoAdquisicion = 1;
led = 1;

fp2 = fopen("/mbedFlash/OrdEns.txt","r");

if (fp2 == NULL) {
                led2 = 1;
                fp2 = fopen("/mbedFlash/OrdEns.txt","w");
fprintf(fp2,"0");
fclose(fp2);
                fp2 = fopen("/mbedFlash/OrdEns.txt","r");

}

fscanf(fp2,"%d",&ContadorEnsayos);
ContadorEnsayos++;

if (ContadorEnsayos == 100) {
ContadorEnsayos = 0;
            }

fclose(fp2);

fp2 = fopen("/mbedFlash/OrdEns.txt","w");
fprintf(fp2,"%d",ContadorEnsayos);
fclose(fp2);

charresultsFile[50];
strcpy(resultsFile,"/local/dat_V");

int n = sprintf(NoViaje,"%d",ContadorEnsayos);
strcat(resultsFile,NoViaje);
strcat(resultsFile,".txt");

            //msc.disk_initialize();

fp = fopen(resultsFile,"w");
if (fp == NULL){
exit(1);
            }
else {
fprintf(fp,"%c Tiempo | aX(m/s2) | aY(m/s2) | aZ(m/s2) | wX(deg/s) | wY(deg/s) |
wZ(deg/s) | pulsosR | pulsosL \n",37);
            }
```

```
        }
else {
tipoAdquisicion = 0;
led = 0;
fclose(fp);
        }
wait(0.5);
      }

GuardaDatosBuffer();
   }
}

voidadquisition(void) {

aX = acc3.getX();
aY = acc3.getY();
aZ = acc3.getZ();

Acel3[0] = ((float)aX - a_xBias3) * 0.002396136798090;
Acel3[1] = ((float)aY - a_yBias3) * 0.002363693255402;
Acel3[2] = ((float)aZ - a_zBias3) * 0.002393961176122;

Gyro3[0] = ((float)gyro3.getGyroX() - w_xBias3) / 14.375;
Gyro3[1] = ((float)gyro3.getGyroY() - w_yBias3) / 14.375;
Gyro3[2] = ((float)gyro3.getGyroZ() - w_zBias3) / 14.375;

pulsos_R = wheel_R.getPulses();
pulsos_L = wheel_L.getPulses();

buff_aX.put(Acel3[0]);
buff_aY.put(Acel3[1]);
buff_aZ.put(Acel3[2]);

buff_wX.put(Gyro3[0]);
buff_wY.put(Gyro3[1]);
buff_wZ.put(Gyro3[2]);

buff_wheel_R.put(pulsos_R);
buff_wheel_L.put(pulsos_L);

}

voidGuardaDatosBuffer() {

bool empty = buff_aX.isEmpty();
intwi = buff_aX.getWritingIndex();
intri = buff_aX.getReadingIndex();
intdif;

if ((wi-ri)>0) {
dif = wi-ri;
   }
else {
dif = sizeBuff+wi-ri;
}
```

```
floattiempo;

if (empty == false) {

tiempo = ((float)ContadorDatos)*((float)PasoTiempo)/1.0e6;

if (tipoAdquisicion == 1) {
fprintf(fp,"%.2f %.3f %.3f %.3f %.3f %.3f %.3f %.2f %.2f
\n",tiempo,buff_aX.get(),buff_aY.get(),buff_aZ.get(),buff_wX.get(),buff_wY.get(),buff_wZ.get(),
buff_wheel_R.get(),buff_wheel_L.get());
      }
ContadorDatos++;

   }

}

voidcalibracion() {

wait(4);

   a_xBias3 = 0.0; a_yBias3 = 0.0; a_zBias3 = 0.0;
   w_xBias3 = 0.0; w_yBias3 = 0.0; w_zBias3 = 0.0;

for (inti = 0; i< 1000; i++) {

aX = acc3.getX();
aY = acc3.getY();
aZ = acc3.getZ();

wX = gyro3.getGyroX();
wY = gyro3.getGyroY();
wZ = gyro3.getGyroZ();

      a_xBias3 += (float)aX; a_yBias3 += (float)aY; a_zBias3 += (float)aZ;
      w_xBias3 += (float)wX; w_yBias3 += (float)wY; w_zBias3 += (float)wZ;

wait(0.01);

   }

   a_xBias3 /= 1000.0; a_yBias3 /= 1000.0; a_zBias3 /= 1000.0;
   w_xBias3 /= 1000.0; w_yBias3 /= 1000.0; w_zBias3 /= 1000.0;

}

voidinicializaIMU() {
gyro3.setLpBandwidth(LPFBW_42HZ);
gyro3.setSampleRateDivider(5); // Hace 1kHz/5, es decir 200Hz
wait_ms(22);

}
```

# Anexo 4

## Simulación cinemática con Maltab

```
% Simulación cinemática Róver 7 coordenadas
clc
clear
closeall

global Param
Param = ParametrosRóver;

Ind = Param.Ind;
Dep = Param.Dep;
nn  = Param.nn;

inputs = importdata('V62_b.mat');
t      = inputs(:,1);
tet3   = inputs(:,2);
dtet3  = inputs(:,3);
ddtet3 = inputs(:,4);
tet4   = inputs(:,5);
dtet4  = inputs(:,6);
ddtet4 = inputs(:,7);

ax = inputs(:,8);
ay = inputs(:,9);
az = inputs(:,10);
wx = inputs(:,11);
wy = inputs(:,12);
wz = inputs(:,13);

At = t(end) - t(end-1);

x2_0   = 0;
y2_0   = 0;
psi2_0 = 0;
psi5_0 = 0;
tet6_0 = 0;

qdep_0  = [x2_0 y2_0 psi2_0 psi5_0 tet6_0]';

q   = zeros(nn,length(t));
dq  = zeros(nn,length(t));
ddq = zeros(nn,length(t));
```

```
for i = 1 : length(t)

t(i)

qind   = [tet3(i) tet4(i)]';
dqind  = [dtet3(i) dtet4(i)]';
ddqind = [ddtet3(i) ddtet4(i)]';

if t(i) == 0.0
qdep   = qdep_0;
end

q(:,i) = ReordenaCoordenadas(qind,qdep);

    BRod    = MatrizB_Rodadura(q(:,i));
    BRodind = BRod(:,Ind);
    BRoddep = BRod(:,Dep);

dqdep    = -BRoddep\(BRodind*dqind);

dq(:,i)  = ReordenaCoordenadas(dqind,dqdep);

dBRod    = dMatrizB_Rodadura(q(:,i),dq(:,i));

ddqdep   = -BRoddep\(BRodind*ddqind + dBRod*dq(:,i));

ddq(:,i) = ReordenaCoordenadas(ddqind,ddqdep);

    dx2   = dqdep(1);   ddx2   = ddqdep(1);
    dy2   = dqdep(2);   ddy2   = ddqdep(2);
    dpsi2 = dqdep(3);   ddpsi2 = ddqdep(3);
    dpsi5 = dqdep(4);   ddpsi5 = ddqdep(4);
    dtet6 = dqdep(5);   ddtet6 = ddqdep(5);

x2   = x2_0   +   dx2*At + 0.5*At^2*ddx2;
    y2   = y2_0   +   dy2*At + 0.5*At^2*ddy2;
    psi2 = psi2_0 + dpsi2*At + 0.5*At^2*ddpsi2;
    psi5 = psi5_0 + dpsi5*At + 0.5*At^2*ddpsi5;
    tet6 = tet6_0 + dtet6*At + 0.5*At^2*ddtet6;

x2_0   = x2;
    y2_0   = y2;
    psi2_0 = psi2;
    psi5_0 = psi5;
    tet6_0 = tet6;

    q(:,i) = [x2 y2 psi2 tet3(i) tet4(i) psi5 tet6]';

qdep = [x2 y2 psi2 psi5 tet6]';

end
```

# Anexo 5

## Simulación dinámica inversa con Maltab

```
% Simulación Dinámica Inversa
% Simulación Dinámica Inversa Róver 7 coordenadas
clc
clear
closeall

global Param
Param = ParametrosRóver;

Ind = Param.Ind;
Dep = Param.Dep;
nn  = Param.nn;

inputs = importdata('V62_b.mat');
t      = inputs(:,1);
tet3   = inputs(:,2);
dtet3  = inputs(:,3);
ddtet3 = inputs(:,4);
tet4   = inputs(:,5);
dtet4  = inputs(:,6);
ddtet4 = inputs(:,7);

ax = inputs(:,8);
ay = inputs(:,9);
az = inputs(:,10);
wx = inputs(:,11);
wy = inputs(:,12);
wz = inputs(:,13);

At = t(end) - t(end-1);

x2_0   = 0;
y2_0   = 0;
psi2_0 = 0;
psi5_0 = 0;
tet6_0 = 0;

qdep_0  = [x2_0 y2_0 psi2_0 psi5_0 tet6_0]';

q   = zeros(nn,length(t));
dq  = zeros(nn,length(t));
ddq = zeros(nn,length(t));

ParM3 = zeros(length(t),1);
ParM4 = zeros(length(t),1);

for i = 1 : length(t)
```

```
t(i)

qind   = [tet3(i) tet4(i)]';
dqind  = [dtet3(i) dtet4(i)]';
ddqind = [ddtet3(i) ddtet4(i)]';

if t(i) == 0.0
qdep   = qdep_0;
end

q(:,i) = ReordenaCoordenadas(qind,qdep);

    BRod    = MatrizB_Rodadura(q(:,i));
    BRodind = BRod(:,Ind);
    BRoddep = BRod(:,Dep);

dqdep    = -BRoddep\(BRodind*dqind);

dq(:,i)  = ReordenaCoordenadas(dqind,dqdep);

dBRod    = dMatrizB_Rodadura(q(:,i),dq(:,i));

    MMF   = MatMasaFuerzas(q(:,i),dq(:,i),0,0);
    M     = MMF(:,1:nn);
    Qv    = MMF(:,nn+1);
    Qgrav = MMF(:,nn+2);

    G3 = MatrizG3(q(:,i));
    G4 = MatrizG4(q(:,i));

    S = [G3(2,:)' G4(2,:)'];

    M_dep = [M(:,1) M(:,2) M(:,3) M(:,6) M(:,7)];
    M_ind = [M(:,4) M(:,5)];

    BRod_dep = [BRod(:,1) BRod(:,2) BRod(:,3) BRod(:,6) BRod(:,7)];
    BRod_ind = [BRod(:,4) BRod(:,5)];

    AA = zeros(12,12);
bb = zeros(12,1);

AA(1:7,1:5)  = M_dep;
AA(1:7,6:7)  = -S;
AA(1:7,8:12) = BRod';

AA(8:12,1:5)  = BRod_dep;
AA(8:12,6:7)  = zeros(5,2);
AA(8:12,8:12) = zeros(5,5);

bb(1:7)  = -M_ind*ddqind + Qgrav + Qv;
bb(8:12) = -dBRod*dq(:,i) - BRod_ind*ddqind;

    x = AA\bb;

ParM3(i) = x(6);
ParM4(i) = x(7);

Lambdas(:,i) = x(8:12);

ddqdep = x(1:5,1);
```

```
    dx2   = dqdep(1);   ddx2   = ddqdep(1);
dy2   = dqdep(2);   ddy2   = ddqdep(2);
    dpsi2 = dqdep(3);   ddpsi2 = ddqdep(3);
    dpsi5 = dqdep(4);   ddpsi5 = ddqdep(4);
dtet6 = dqdep(5);   ddtet6 = ddqdep(5);

x2   = x2_0   +   dx2*At + 0.5*At^2*ddx2;
    y2   = y2_0   +   dy2*At + 0.5*At^2*ddy2;
    psi2 = psi2_0 + dpsi2*At + 0.5*At^2*ddpsi2;
    psi5 = psi5_0 + dpsi5*At + 0.5*At^2*ddpsi5;
    tet6 = tet6_0 + dtet6*At + 0.5*At^2*ddtet6;

x2_0   = x2;
    y2_0   = y2;
    psi2_0 = psi2;
    psi5_0 = psi5;
    tet6_0 = tet6;

    q(:,i) = [x2 y2 psi2 tet3(i) tet4(i) psi5 tet6]';

qdep = [x2 y2 psi2 psi5 tet6]';

end
```